CICS/VSE Client/Server Solutions
Implementing the Message Queue Interface

Document Number GG24-4263-00

July 1994

International Technical Support Organization
Boeblingen Center

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xv.

First Edition (July 1994)

This edition applies to

- ezBRIDGE Transact on VSE/ESA for IBM MQSeries, Program Number 5787-ECX for use with the VSE/ESA* operating system
- ezBRIDGE Transact on OS/2 for IBM MQSeries, Program Number 5787-EDD for use with the OS/2* operating system
- ezBRIDGE Transact on AIX/6000 for IBM MQSeries, Program Number 5787-ECY for use with the AIX* operating system

The ezBRIDGE** Transact** products are part of the IBM MQSeries family of products which implement the messaging and queuing technology using the Message Queue Interface (MQI).

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 3222 Building 71032-02
Postfach 1380
71032 Boeblingen, Germany

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Abstract

This document describes how to implement applications that use distributed processing techniques based on message queuing techniques provided by IBM′s MQSeries* family of products. Although MQSeries products are available on various operating system platforms the scope of this document is limited to message queuing between VSE/ESA, AIX and OS/2 operating systems. For implementation guidelines on MVS* and OS/400* refer to *Examples of Using MQSeries on S/390, RISC System/6000, AS/400 and PS/2, GG24-4326*, published by the ITSO Raleigh Center.

This publication is intended for system engineers or programmers responsible for implementing distributed applications using message queuing techniques on subject platforms. The reader is assumed to have a basic knowledge of IBM′s Message Queue Interface (MQI) concepts and a working knowledge of the pertinent operating system platforms. In addition, he should be familiar with CICS/VSE* and SNA APPC/LU 6.2 communication techniques which are required to connect the different platforms reflected in this document.

(169 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help Systems Engineers, Marketing Representatives and Customers to implement Client/Server solutions using messaging and queuing techniques offered by IBM′s MQSeries family of products.

The document describes how to install and customize the corresponding MQSeries product on each of the three platforms involved, that is VSE/ESA for the host system, OS/2 for PS/2* and AIX for RISC System/6000* workstations. In addition, the SNA-based communication between the host and the workstations and the implementation of a sample application are explained.

The information in this publication is not intended as the specification of any programming interfaces that are provided by components of various solutions discussed here. See the PUBLICATIONS section of the IBM Programming Announcement for respective products described in this document for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT 06904 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AIX/6000
ACF/VTAM	AS/400
CICS/VSE	IBM
MQSeries	OS/2
OS/400	PROFS
PS/2	RISC System/6000
System/88	System/370
System/390	VM/ESA
VSE/ESA	VTAM

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

ezBRIDGE	System Strategies, Inc.
HP and HP-UX	Hewlett Packard Company
IEEE	Institute of Electrical and Electronics Engineers
Macintosh	Apple Computer Inc.
Micro Focus	Micro Focus Ltd.
Microsoft, Windows	Microsoft Corporation
Motif	Open Software Foundation, Inc.
Network File System and NFS	SUN Microsystems, Inc.
NetWare	Novell, Inc.
Novell	Novell, Inc.
POSIX	IEEE
SPARC and Solaris	SUN Microsystems, Inc.
SUN	SUN Microsystems, Inc.
Tandem, Guardian, NonStop	Tandem Computers, Inc.
Transact	System Strategies, Inc.
UNIX	X/Open Company Ltd.
VMS	Digital Equipment Corporation

Preface

The purpose of this document is to help IBM personnel and/or customers to implement *Client/Server Computing* solutions that are based on the message queuing capabilities of IBM′s MQSeries family of products. It contains a description of the installation, customization and implementation steps required to implement applications running on VSE/ESA, OS/2 and AIX using the corresponding MQSeries product.

This document is intended for persons who want to implement distributed applications using the Message Queue Interface (MQI) on VSE/ESA hosts and PS/2 or RISC System/6000 workstations. It complements the ITSO Raleigh Center publication *Examples of Using MQSeries on S/390, RISC System/6000, AS/400 and PS/2, GG24-4326* which provides equivalent information for MVS and AS/400* platforms (refer to “Related Publications” on page xviii).

How This Document is Organized

The document is organized as follows:

- Chapter 1, “VSE/ESA Client/Server Computing using Messaging and Queuing”

  This chapter gives a brief introduction of the concepts and architecture of Messaging and Queuing.

- Chapter 2, “Overview: Message Queuing between VSE/ESA and OS/2”

  This chapter introduces the functions and facilities for distributed applications using MQI between OS/2 workstations and VSE/ESA.

- Chapter 3, “The MQI Test Environment”

  This chapter describes our hardware and software environment, shows the network configuration we used and illustrates the MQI scenarios we implemented.

- Chapter 4, “ezBRIDGE on VSE/ESA Implementation”

  This chapter provides the definitions required in VSE/ESA to implement our distributed MQI application.

- Chapter 5, “ezBRIDGE on OS/2 Implementation”

  This chapter describes how to implement MQI on OS/2 workstations.

- Chapter 6, “MQI Applications for OS/2 and VSE/ESA”

  This chapter provides guidelines on how to implement distributed MQI applications between a VSE/ESA host and OS/2 workstations.

- Chapter 7, “ezBRIDGE on VSE/ESA to ezBRIDGE on AIX Overview”

  This chapter introduces the functions and facilities for distributed applications using MQI between AIX workstations and VSE/ESA.

- Chapter 8, “ezBRIDGE on VSE/ESA to ezBRIDGE on AIX Test Environment”

  This chapter describes our hardware and software environment, shows the network configuration we used and illustrates the MQI scenario we implemented.

- Chapter 9, “Connecting the AIX Workstation to the Host”

  This chapter describes how MQI AIX workstations and the VSE/ESA host are connected to each other.

- Chapter 10, “ezBRIDGE on AIX Implementation”

  This chapter describes how to implement MQI on AIX workstations.

- Chapter 11, “MQI Applications for AIX and VSE/ESA”

  This chapter provides guidelines on how to implement distributed MQI applications between a VSE/ESA host and AIX workstations.

- Appendix A, “ezBRIDGE on VSE/ESA Sample Definitions”

  This chapter provides jobstreams and definitions used during implementation of ezBRIDGE on VSE/ESA.

- Appendix B, “ezBRIDGE on AIX Sample Definitions”

  This chapter provides jobstreams and definitions used during implementation of ezBRIDGE on AIX.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

ezBRIDGE Publications

- *MQSeries: An Introduction to Messaging and Queuing, GC33-0805*
- *Messaging and Queuing Series Technical Reference, SC33-0850*
- *ezBRIDGE Transact for MQSeries Overview, Concepts and Architecture, GC33-1141*
- *ezBRIDGE Transact for VSE/ESA for MQSeries User Manual, SC33-1142*
- *ezBRIDGE Transact on OS/2 for MQSeries User Manual, SC33-1148*
- *ezBRIDGE Transact on AIX/6000 for MQSeries User Manual, SC33-1143*

Other IBM Publications

- *AIX/SNA Server/6000: User*
- *CICS/VSE System Programming Reference, SC33-0711*
- *VTAM Messages and Codes, SC31-6433*
- *VTAM Operation, SC31-6435*
- *VTAM Resource Definition Reference, SC31-6438*
- *VTAM Network Implementation Guide, SC31-6434*
- *SNA Network Product Formats, LY43-0081*

International Technical Support Organization Publications

- *Examples of Using MQSeries on S/390, RISC System/6000, AS/400 and PS/2, GG24-4326*
- *Messaging and Queuing Extensions for VSE/ESA, GG24-4296*
- *TCP/IP Solutions for VSE/ESA, Implementation Guide, GG24-4195*

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

To get listings of redbooks online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Technical Bulletins (Redbooks)

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their IBM branch office.

Customers may order hardcopy redbooks individually or in customized sets, called GBOFs, which relate to specific functions of interest. You may also order redbooks in online format on CDROM collections, which contain the redbooks for multiple products.

Acknowledgments

The advisor for this project was:

Werner Stieber
International Technical Support Organization, Center Boeblingen

The authors of this document are:

Ken Mitchell
IBM Canada

Mark Toplak
IBM Slovenia

Norbert Wieland
IBM Germany

This publication is the result of a residency conducted at the International Technical Support Organization, Center Boeblingen.

Thanks to the following people for their advice and guidance provided in the production of this document:

John Bennett
ITSO Center Boeblingen

Ian Craggs
IBM Laboratories Hursley, UK

Yukihiko Yoshida
ITSO Center Boeblingen

Part 1. Introduction

This document describes the communication aspects of implementing client/server solutions which use IBM′s Message Queue Interface (MQI) techniques between VSE/ESA, OS/2 and AIX platforms.

The introductory part provides a brief description of MQI concepts and a list of IBM′s MQSeries family of products.

Chapter 1. VSE/ESA Client/Server Computing using Messaging and Queuing

1.1 IBM′s Message Queue Interface: Basic Concepts

The Message Queue Interface (MQI) is one of the three most prevalent network communication styles besides CPI-C and RPC and is part of IBM′s Networking Blueprint introduced in March 1992. It is a method of program-to-program communication suitable for connecting independent and potentially non-concurrent distributed applications.

The concept of message queuing is that participating applications exchange data, that is communicate with each other via *Message Queues*. Thus the basic elements of this technique are:

- **Messages** - Strings of bits and bytes that have a meaning to the participating programs, that is application data; control information is added for controlling storage, routing and delivery of the message.
- **Message Queues** - Named ″objects″ where messages are stored by an application program, and from which they are later retrieved (typically) by another application program. A message queue always belongs to a **Queue Manager**, a system service whose main functions are to:
  - manage the queues for application programs
  - provide an application programming interface, that is the Message Queue Interface (MQI)
  - transfer messages to another (typically remote) queue manager via **message channels**

  Queues are called **local** if they are owned by the queue manager of the same (local) system. Queues belonging to another system are called **remote** queues. Local queues holding messages that are to be forwarded to a remote queue manager are called **transmission queues**.

Messages are transferred between queues in one direction only using a uni-directional point-to-point communication link called a **message channel**. Message channels are started and controlled by a **message channel agent**, an executable program which implements a specific **Message Channel Protocol (MCP)**. The MCP in turn uses the industry standard transport protocols SNA or TCP/IP.

Figure 1 on page 4 illustrates the basic concepts discussed above by showing a distributed queuing example between a VSE/ESA host and an AIX workstation:

- Message queuing dataflow on the VSE/ESA host:
  - Application Program A puts a message into queue TOAIX
  - TOAIX is a remote queue for queue manager QMVSE on VSE/ESA
  - TOAIX uses transmission queue QMAIX to route messages to the remote queue manager QMAIX
  - QMAIX uses message channel CVSEAIX to transfer the message to AIX. CVSEAIX uses SNA APPC/LU 6.2 protocols to connect to AIX

- Message queuing dataflow on the AIX workstation:
  - The message arrives via message channel CVSEAIX and is routed to FROMVSE
  - FROMVSE is a local queue for queue manager QMAIX on AIX (queue manager QMVSE knows it as TOAIX)
  - Application Program B gets the message from its local queue named FROMVSE

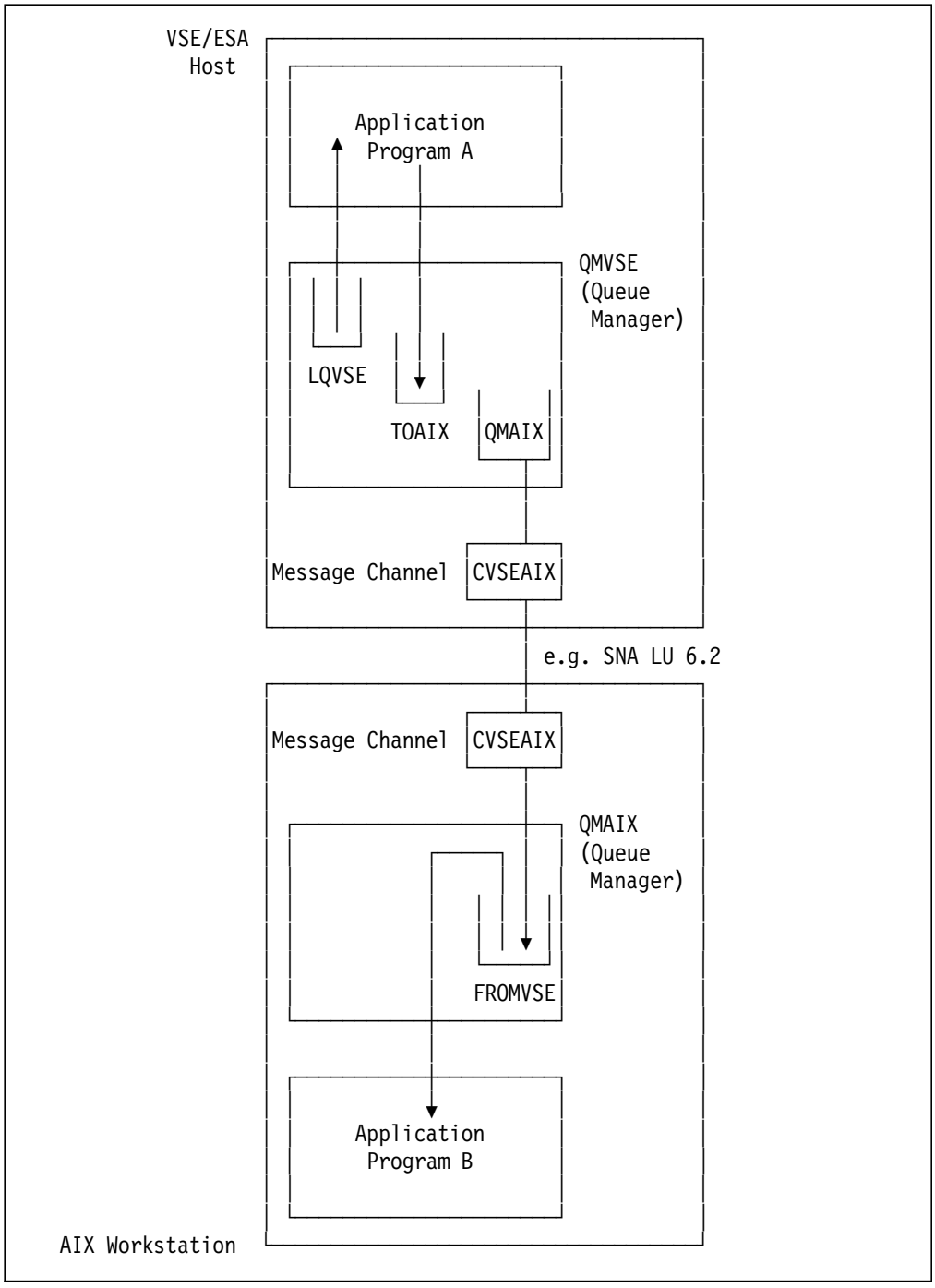

Figure 1. Basic MQI Elements and Definitions

1.2 The MQSeries Product Family

The MQSeries family of products provides for MQI implementations on IBM platforms. In addition, there are MQI products on systems from other manufacturers. IBM′s MQSeries products include:

- Message Queue Manager MVS/ESA* (MQM MVS/ESA), which implements MQI for
  - CICS*
  - IMS
  - TSO/E
  - Batch Environments
- Message Queue Manager 400 (MQM/400), which implements MQI for applications running in the AS/400 system
- ezBRIDGE Transact for IBM MQSeries products on the following platforms:
  - Personal computer systems DOS**, Windows**, or OS/2
  - IBM* System/88* (OS/88)
  - IBM AS/400 (OS/400)
  - IBM RISC System/6000
  - IBM System/370* and System/390* mainframes with CICS/VSE*

Queue managers on systems from other manufacturers:

- DEC**VAX (VMS)
- HP/UNIX** (HP/UX)
- Santa Cruz** Operations (SCO)
- Sun** Microsystems (SunOS)
- Tandem** NonStop (Guardian 90)
- Unixware

This document provides information on the MQI implementation for VSE/ESA, AIX and OS/2 platforms, that is

- ezBRIDGE Transact on VSE/ESA for IBM MQSeries
- ezBRIDGE Transact on AIX/6000 for IBM MQSeries
- ezBRIDGE Transact on OS/2 for IBM MQSeries

Equivalent information for MVS and AS/400 platforms can be found in *Examples of Using MQSeries on S/390, RISC System/6000, AS/400 and PS/2, GG24-4326*.

Part 2. MQI Client/Server Implementation between VSE/ESA and OS/2

Chapter 2. Overview: Message Queuing between VSE/ESA and OS/2

The next four chapters document the parameters used to customize the environment described in Figure 2 on page 12 which provides for message queuing between the VSE/ESA host and two OS/2 workstations. Implementation of IBM′s Message Queue Interface protocol is handled by different products on different platforms. The following sections summarize the features of the protocol′s implementation on VSE/ESA and OS/2 platforms and highlight some of the differences.

2.1 ezBRIDGE on VSE/ESA

ezBRIDGE Transact on VSE/ESA for IBM MQSeries is a member of the IBM MQSeries family of products which allows CICS/VSE applications to use the **Message Queue Interface (MQI)** to send messages to other applications. These other applications may reside on the same or on a different CICS/VSE system, or on any other platform which supports the message queue interface.

CICS/VSE application programs may write messages on ′Queues′ which reside on VSAM files on the local CICS/VSE system or they may be located on a remote message queue system. The queues may be defined in such a way that the location is transparent to the application program.

On VSE/ESA, the MQI is built around the standard COBOL function call interface. COBOL II (Release 3 or higher) is required.

The transportation function used to send messages between Transact systems is handled by a **Message Channel Agent (MCA)**. On the VSE/ESA platform, this MCA is a set of CICS transactions that use SNA LU 6.2 protocols.

As with all other Transact environments, there is a set of **Message Queue Management (MQM)** functions for administration and operation. These are provided on the CICS/VSE platform as another group of CICS transactions.

For detailed information on ezBRIDGE on VSE/ESA refer to the ITSO Center Boeblingen publication *Messaging and Queuing Extensions for VSE/ESA, GG24-4296*.

2.2 ezBRIDGE on OS/2

ezBRIDGE Transact on OS/2 for IBM MQSeries is a member of the IBM MQSeries family of products which allows OS/2 applications to use the Message Queue Interface (MQI) to send messages to other applications. These other applications may reside on the same or on a different OS/2 system, or on any other platform which supports the message queue interface.

OS/2 or DOS application programs may write messages on ′Queues′ which may reside on the local Personal Computer file system, on a LAN server′s file system, or they may be located on a remote message queue system. As in the VSE/ESA ezBRIDGE environment, the location may be defined in such a way as to be transparent to the application program. If a common file system exists, for example in a LAN file server environment, Transact views it as a single system regardless of the number of machines which are served by it. In our scenario,

the two OS/2 systems **do not share files** via a LAN server. From a Transact point of view they are considered separate systems, each with its own unique Queue Manager.

On OS/2, the MQI is built around the standard C language function call interface. Application development for the OS/2 environment requires the IBM C set++ version 2.0 language compiler. Application development for the DOS environment requires the Microsoft C/C++ version 7 language compiler.

The transportation function used to send messages between Transact systems is handled by a **Message Channel Agent (MCA)**. On the OS/2 platform, this MCA is a set of programs which use either SNA LU 6.2, or TCP/IP protocols.

The **Message Queue Management (MQM)** functions for administration and operation are provided as a menu-driven program called ***MQM***.

2.3 Differences between ezBRIDGE on VSE/ESA and ezBRIDGE on OS/2

Some of the differences between the OS/2 and VSE/ESA implementations of Transact which are noteworthy in our scenario are:

- Applications distributed over our two platforms will require programming skills in two languages. COBOL for the CICS/VSE environment and C for the PC environment.
- Personal systems store data in **ASCII** format. VSE/ESA uses **EBCDIC**. At the time of writing this document, neither of these two ezBRIDGE implementations provided facilities for message data conversion from one platform to another. Conversion of message data is the responsibility of the application program.
- The VSE/ESA implementation supports **triggers** which allow the automatic initiation of a program upon receipt of a message in a particular queue. Only pseudo-triggering is supported by ezBRIDGE on OS/2. With pseudo-triggering, an application must issue the message queue ′GET′ function and then wait until a message arrives or the WaitInterval expires before regaining control. Thus OS/2 programs which read from Transact queues must be started externally.
- The OS/2 implementation supports SNA LU 6.2 as well as TCP/IP protocols, whereas ezBRIDGE on VSE/ESA supports SNA LU 6.2 only.

Chapter 3. The MQI Test Environment

This chapter describes the hardware and software used in our project and illustrates the MQI scenario between OS/2 and VSE/ESA.

Our environment consists of:

- a VSE/ESA host running ezBRIDGE Transact on VSE/ESA for IBM MQSeries
- two workstations running ezBRIDGE Transact on OS/2 for IBM MQSeries

The workstations running ezBRIDGE on OS/2 and the ezBRIDGE on VSE/ESA host are connected to each other via an IBM Token-Ring LAN running SNA protocols. The network is illustrated in Figure 2 on page 12.

3.1 Hardware

- An IBM 9221 Model 150 as the host with:
  - 64MB main memory
  - Token-Ring connection via channel attached IBM 3172
- Two IBM PS/2 Model 90 workstations running ezBRIDGE on OS/2

3.2 Software

- VM/ESA* R2.1 with GCS and ACF/VTAM V3.4.1 in the IBM 9221
- VSE/ESA Version 1.3.3 guest under VM in the IBM 9221 with
  - ACF/VTAM V3.4.0*
  - CICS/VSE V2.2
  - ezBRIDGE Transact on VSE/ESA for IBM MQSeries R3.0
  - VS COBOL II Compiler and Libraries 1.4.0
- Interconnect Controller Program (ICP) Version 3.2 in the IBM 3172 Model 3
- OS/2 Version 2.1 running in the IBM PS/2 workstations with
  - CM/2 V1.0
  - ezBRIDGE Transact on OS/2 for IBM MQSeries R3.0

3.3 Network Configuration

Figure 2 on page 12 illustrates the network described above including the MAC-addresses used in the appropriate VTAM, CM/2 and 3172 ICP definitions.

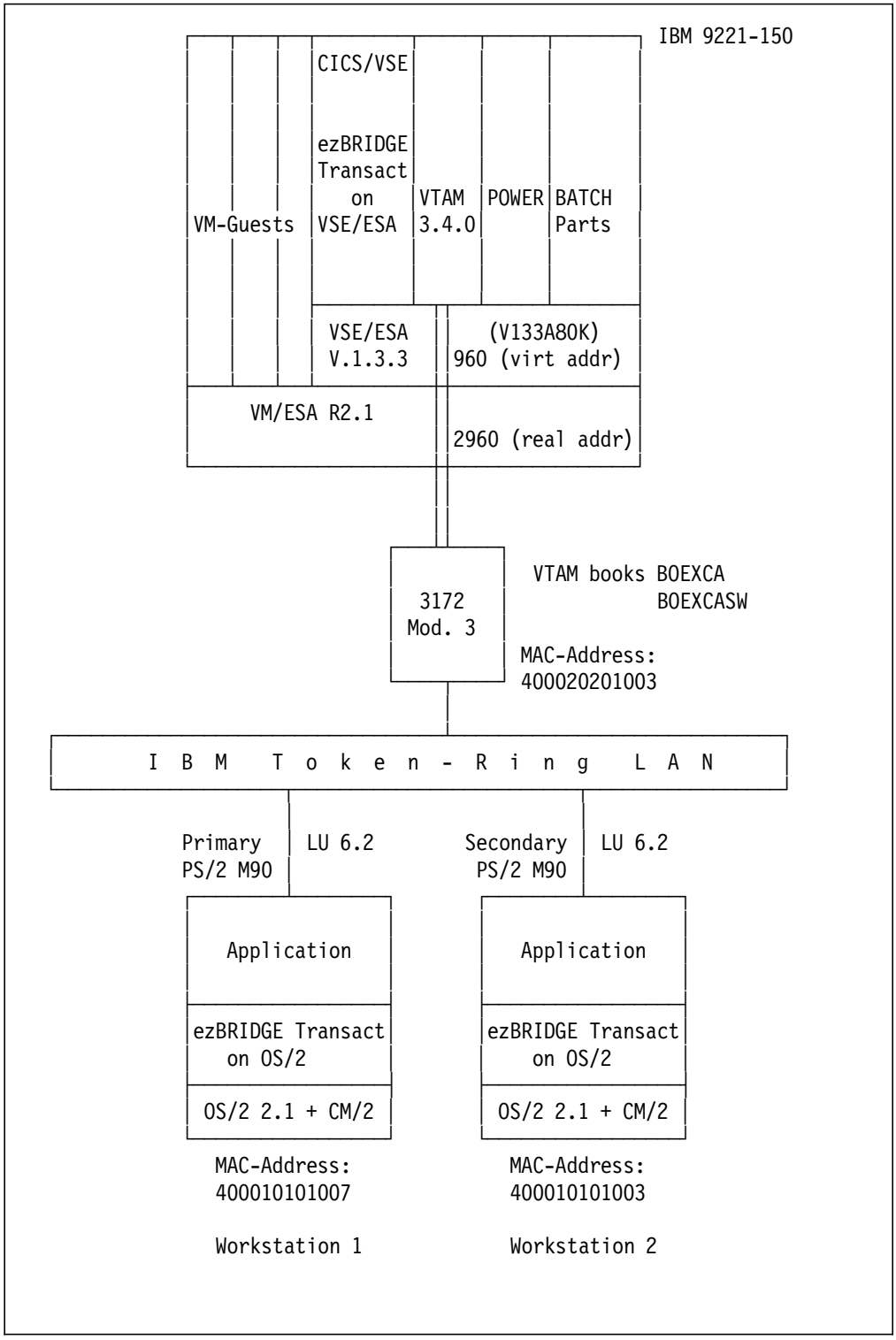

Figure 2. ezBRIDGE on VSE/ESA to ezBRIDGE on OS/2 MQI Network Diagram

3.4 ezBRIDGE on VSE/ESA to ezBRIDGE on OS/2 MQI Examples

Figure 3 on page 15 illustrates a distributed MQI example for message queuing between a VSE/ESA host and two OS/2 systems. It shows the interrelations of message queues, message channels and how the systems are connected to each other. This diagram should be referenced while studying the sample definitions provided in this document.

Figure 3 on page 15 shows ezBRIDGE on VSE/ESA connected to two ezBRIDGE on OS/2 systems:

- one on OS/2 Workstation 1, named **OS2S**
- one on OS/2 Workstation 2, named **OS22**

The ezBRIDGE-Systems on workstations 1 and 2 are in turn connected to each other. For this connection two channels are defined on each system, one for sending and one for receiving messages.

First let us follow the flow of a message sent from an application on VSE/ESA to a queue on workstation 1 (OS2S):

1. An application program writes a message to the queue **TOOS2** by using the supplied MQI function MQPUT. TOOS2 is, from the host (VSE/ESA) point of view, a ′remote′ queue, for which an associated transmission queue named ′**QMOS2S**′ has been defined.
2. The message is physically stored on QMOS2S.
3. The message is picked up by the host′s MCA and sent to queue **FROMVSE**, a local queue on OS2S using an **APPC connection** between message channels **CVSEOS2** on VSE/ESA and **CVSEOS2** on OS/2 (equal names on both platforms were chosen for the message channels in order to indicate the message flow).
4. The connection will be established when the sending channel requests an ′APPC attach′ from the receiver side, where CM/2 starts the MCA via the transmitted TP program name.
5. As soon as the receiving MCA is operational, the incoming message is stored on queue **FROMVSE**. From there an application can read the message using MQGET.

The reverse message flow from OS2S to the host is quite similar:

1. The message is written to remote queue **TOVSE** and stored on transmission queue **QMVSE**.
2. OS2S′s MCA sends the message to the host′s local queue **FROMOS2** using the APPC connection between message channels **COS2VSE** on both sides.
3. The receiving MCA on VSE/ESA is activated via TRANS-ID ′MQ01′ provided by the sending MCA on OS/2.
4. Host applications are now able to read the message from FROMOS2 using MQGET.

Now it is easy to follow the message flow between the two workstations, that is **OS2S** and **OS22**:

1. From OS2S to OS22:
   a. MQPUT the message to remote queue **TOOS22** on OS2S.

b. Send the message to OS22 via transmission queue **QMOS22** using message channels **COS2SOS22**.

c. The receiving MCA on OS22 will be started by OS/2′s CM/2 and put the message into OS22′s local queue **FROMOS2S**.

2. From OS22 to OS2S:

a. MQPUT the message to remote queue **TOOS2S** on OS22.

b. Send the message to OS2S via transmission queue **QMOS2S** using message channels **COS22OS2S**.

c. The receiving MCA on OS2S will be started by OS/2′s CM/2 and put the message into OS2S′s local queue **FROMOS22**.

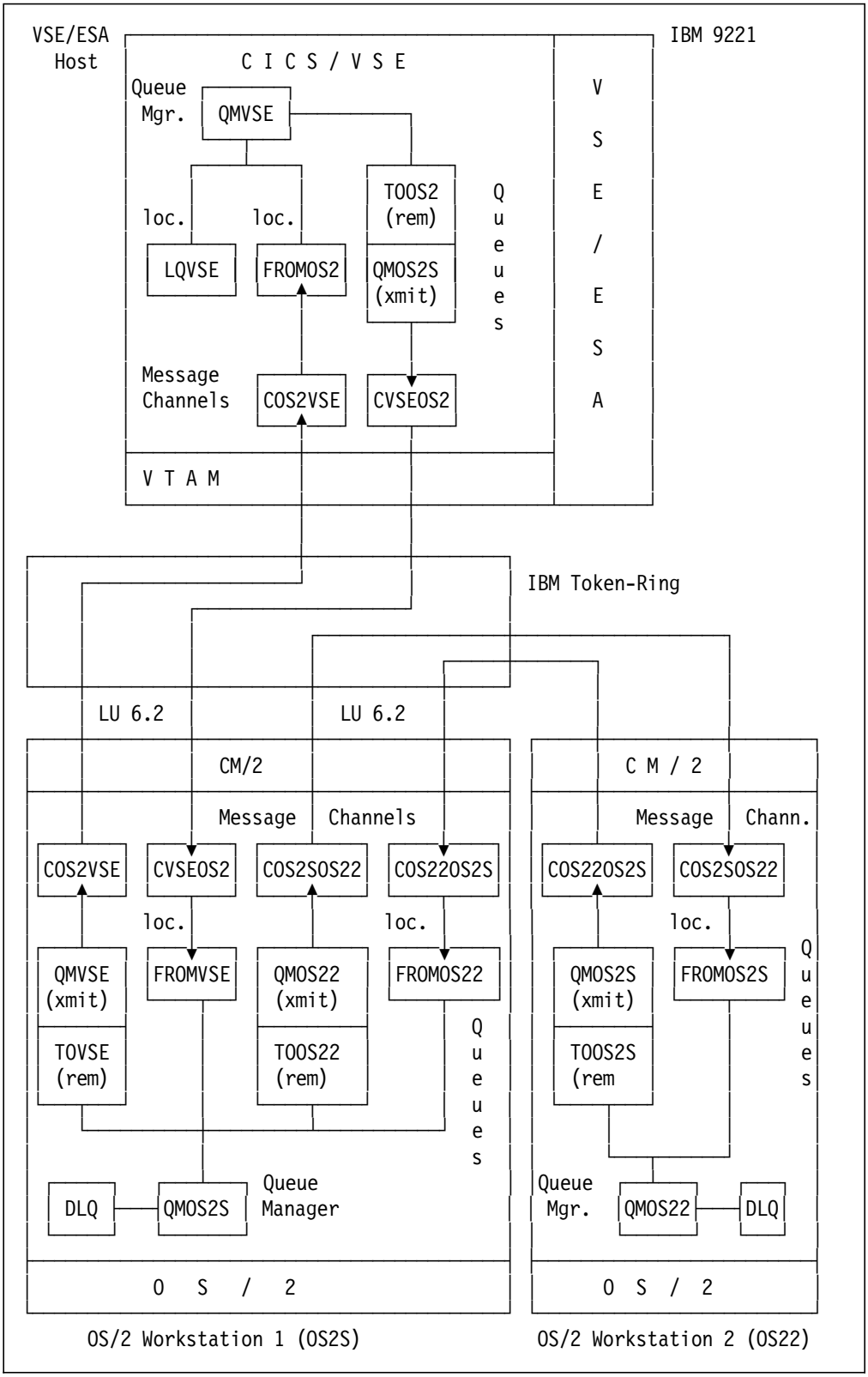

Figure 3. Message Queuing between VSE/ESA - Workstation 1 (OS2S) and Workstation 1 - Workstation 2 (OS22)

Figure 4 on page 17 illustrates a distributed MQI example for message queuing between a VSE/ESA host and an OS/2 system (OS22) using another OS/2 queue manager (on OS2S) as ′Message Router′. It shows the interrelations of message queues, message channels and how the systems are connected to each other. This diagram should be referenced while studying the sample definitions provided in this document.

The diagram shows that ezBRIDGE on VSE/ESA is not directly connected to ezBRIDGE on workstation 2 (OS22), but via ezBRIDGE on workstation 1 (OS2S). All three MQI systems are attached to the same Token-Ring, but there is no direct exchange of messages between VSE/ESA and OS22.

Let us follow the flow of a message sent from an application on VSE/ESA to queue FROMVSE on workstation 2.

1. An application program writes a message to queue **TOOS22** using MQPUT. This remote queue TOOS22 feeds the transmission queue **QMOS2S**. The definitions for TOOS22 specify queue manager **QMOS22** on OS22 as the target queue manager and **FROMVSE** as the remote queue where the message should arrive.
2. The message is stored in transmission queue QMOS2S.
3. The hosts′s MCA sends the message to queue FROMVSE on OS2S via message channels CVSEOS2 defined on VSE/ESA and OS2S.
4. The supplied queue manager name ′QMOS22′ (destination) causes the message to be transferred to the transmission queue **QMOS22** on OS2S.
5. OS2S′s MCA sends the message to its final destination, FROMVSE on OS22 using the message channels COS2SOS22 defined on both workstations.
6. Applications on OS22 can now read the message using MQGET.

A possible analogy for this process might be a suitcase (message) on its way from the airport where it was checked-in (VSE/ESA) to its final destination airport (OS22). The ′flow′ of the suitcase is controlled by a tag wrapped around the handle containing the destination code (the target queue manager specification ′QMOS22′).

The reverse message flow from OS22 to VSE/ESA is similar:

1. The definition of an additional remote queue **TOVSE** on workstation 2 (OS22) points to the transmit queue **QMOS2S**.
2. A message put to TOVSE by MQPUT is written to queue QMOS2S.
3. OS22′s MCA sends the message to queue FROMOS22 on OS2S using the message channels COS22OS2S defined on both workstations.
4. OS2S′s MCA sends the message to its final destination, FROMOS2 on the host via transmission queue QMVSE and by using the message channels COS2VSE defined on VSE/ESA and OS2S.

   When the message is sent via message channels COS2VSE, a four-letter trans-id (**MQ01**) is supplied to start the receiver task (the MCA) for COS2VSE under CICS/VSE.

We did not define an extra queue on VSE/ESA to differentiate from where the messages come. Related to Figure 4 on page 17 this means that queue FROMOS2 on VSE/ESA receives messages from both systems, OS2S and OS22.

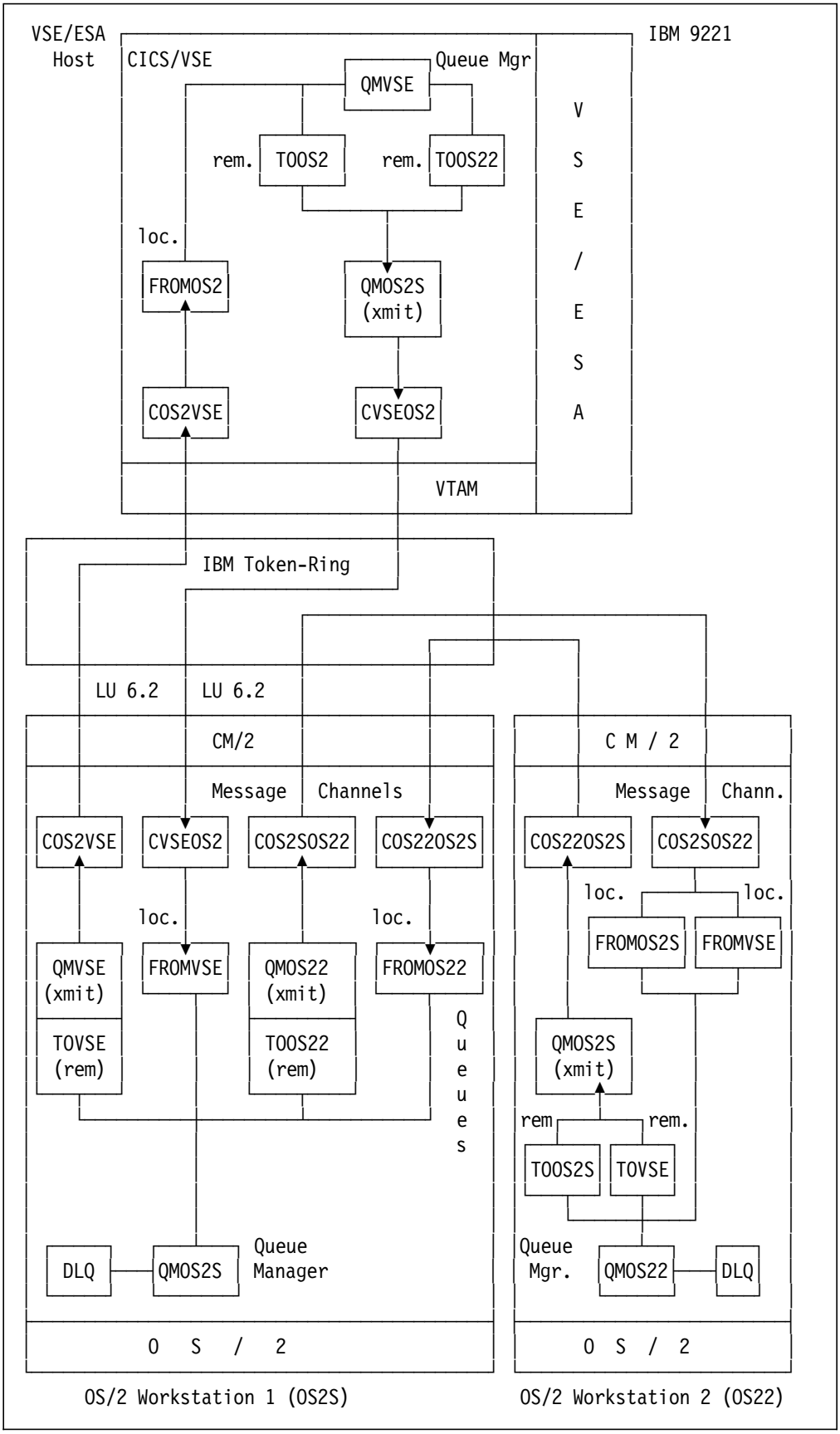

Figure 4. Message Queuing between VSE/ESA and Workstation 2 (OS22) via Workstation 1 (OS2S)

Based on the configuration shown in figure 3 we were able to send a message from system 1 (OS2S) to system 2 (OS22) where it was returned automatically. For this we used the following command:

zmqwrite QMOS22#TOOS2S 1 30 This message returns to OS2S.

This starts the program ′***zmqwrite***′ which in this case will send the message ″This message returns to OS2S.″ to local queue ***TOOS2S*** on the system where queue manager ***QMOS22*** resides. The message is immediately returned to queue ***FROMOS22*** on system OS2S, since every message in queue TOOS2S on OS22 is sent to OS2S by OS22′s sender MCA.

Chapter 4. ezBRIDGE on VSE/ESA Implementation

This chapter describes the definitions required in **VM, VSE, CICS/VSE, VSE/VTAM** and **ezBRIDGE** to implement our MQI-based network shown in Figure 2 on page 12 and Figure 3 on page 15.

The network consists of three nodes which communicate with each other via message queuing:

1. The ′MQI host′, that is VSE/ESA running ezBRIDGE on VSE/ESA.
2. A **primary** OS/2 system, named ′OS2S′ (MAC-ADDR 400010101007), defined to VSE/VTAM as an **independent LU**, or Physical Unit (PU) Type 2.1 in order to support APPC parallel sessions between ezBRIDGE platforms.
3. A **secondary** OS/2 system, named ′OS22′ (MAC-ADDR 400010101003), which is configured to communicate directly to the primary OS/2 system via OS/2 Communications Manager. It was **not** configured to communicate ′directly′ to the host ezBRIDGE system. Messages destined for the host were routed through the primary OS/2 system′s Queue Manager to the host′s Queue Manager. Therefore, OS22 is not reflected at all in the host′s VM or VSE/ESA configurations. The only references on the host to the secondary OS/2 system are in the host′s ezBRIDGE remote queue definitions described in chapter 4.4, “ezBRIDGE on VSE/ESA Customization” on page 37.

4.1 ezBRIDGE on VSE/ESA Implementation Overview

The following steps are required to implement a VSE/ESA ′MQI host′ as shown in Figure 2 on page 12 and Figure 3 on page 15:

1. Define the host′s network access via the IBM 3172 Interconnect Controller
2. Provide the appropriate VSE/VTAM definitions for CICS/VSE
3. Customize CICS/VSE, since ezBRIDGE on VSE/ESA is running under control of CICS/VSE
4. Install and customize ezBRIDGE on VSE/ESA

4.2 Network Access via IBM 3172

In our configuration the VSE virtual machine named ′*V133A80K*′ owns the IBM 3172.

The following must be prepared to permit VSE/ESA to control the 3172:

1. The 3172 must be defined to VM/ESA.
   - In our case, this is done via the 9221 machine′s **IOCDS** and VM/ESA R2.1′s automatic device sensing and dynamic configuration capability.
2. The 3172 must be defined to the VSE machine. This involves
   - An appropriate DEDICATE statement in the VM directory for V133A80K.
   - A change in the **VSE IPL** procedure.
3. The 3172 **Interconnect Control Program** (ICP) must be configured to reflect its operating environment.

- In our case this is done with the customization parameters found in Figure 9 on page 22.

4. **VSE/VTAM** books and tables must be created to describe the 3172 and Token-Ring devices.
   - In our case by creating the VTAM books BOEXCA, BOEXCASW, and the LOGMODE entry LU62PS.

4.2.1 VM/ESA Definitions for the IBM 3172

4.2.1.1 IBM 3172 Definition in IBM 9221 IOCDS

The 3172 is a channel attached device and is defined in the 9221 IOCDS (I/O Configuration Data Set).

The following entries in the IOCDS describe the 3172:

```
CHPID PATH=((29)),TYPE=BL
CNTLUNIT CUNUMBR=3172,PATH=(29),UNIT=3172,UNITADD=((60,32)),          X
               SHARED=N,PROTOCOL=S4
IODEVICE ADDRESS=(2960,32),CUNUMBR=(3172),UNIT=3172
```

Figure 5. 3172 IOCDS Entries

4.2.1.2 IBM 3172 Definitions to VM/ESA

VM/ESA R2.1 uses dynamic configuration and automatic device sensing. Static device definition is not required. The statements shown in Figure 6 are included in the VM SYSTEM CONFIG file, to activate and auto-sense device addresses 0000-FFFF during system initialization.

```
Devices,
   Online_at_IPL 0000-FFFF,
   Sensed        0000-FFFF
```

Figure 6. VM/ESA Auto-Sense Definitions in SYSTEM CONFIG File

4.2.2 IBM 3172 Definitions for VSE/ESA

4.2.2.1 Attach the IBM 3172 to the VSE/ESA Machine

For the 3172 to be owned by VSE/ESA, it must be attached or dedicated to the VSE/ESA virtual machine. In our environment this was accomplished with the **DEDICATE** statement as in Figure 7 on page 21 where **2960** is the real address and **960** the virtual address.

```
USER V133A80K PASSWORD 0032M 64M G
  ACCOUNT V133A80K V133A80K
  OPTION MAXCON 150
  MACHINE ESA
  CONSOLE 0009 3215 T
         .
         .
  DEDICATE 960 2960
         .
         .
  MDISK 191 3380  001  049 DISK01 MW ALL
  LINK MAINT   190 0190 RR
  MDISK 991 3380 1770  885 DISK01 MW RPASS WPASS MPASS
```

Figure 7. VM Directory Entry for VSE/ESA Machine

4.2.2.2 Define the IBM 3172 in VSE/ESA

The 3172 is **ADD**ed via the IPL procedure as in Figure 8. VSE/ESA is to treat it as a channel to channel adapter (CTCA). The device is not to be sensed at IPL time (EML). This guarantees that the device characteristics implied by CTCA are not changed dynamically at IPL time by VSE/ESA.

```
   CATALOG $IPLBOE.PROC REPLACE=YES
009,$$A$SUPX,VSIZE=120M,VIO=512K,VPOOL=128K,LOG
ADD 009,3277
ADD 00C,2540R
       .
       .
       .
ADD 960,CTCA,EML        3172 GATEWAY
       .
       .
ADD FFF,CONS            DUMMY CONSOLE, DO NOT DELETE
       .
       .
SVA SDL=300,GETVIS=768K,PSIZE=640K
/+
```

Figure 8. VSE IPL Procedure Including the IBM 3172

4.2.3 IBM 3172 Customization

Parameters used in the customization of the Interconnect Control Program (ICP) for the 3172 are described in Figure 9 on page 22. There are two parameters here which must match definitions elsewhere in our example:

- ***Subchannel.*** This value must match the control unit and unit portion of the real address defined to the host for the 3172. In our case the real address 29**60** is defined to VM/ESA
- ***Node address.*** This value is the **MAC Address** used by OS/2's Communication Manager to find the 3172 gateway to our CICS/VSE host.

```
3172-3 Configuration Summary

 3172 Name ................... : IS23172
 3172 Type ................... : 3172-3 LAN Gateway
 Int Enhancement Feature (IEF) : Yes
 User Data ................... : LAN Gateway for 9221-150(IS2)
 Location .................... : Building 02 Room 018

 ICP Base Code Version........ : 3.02.00
 ICP IEF Code Version......... : 3.02.00
 APARs/Patches applied........ : None

 Profile Name ................ : TRL3172

 Slot    Name        Adapter Type
 ----    ----------  ----------------------
  1      Unassigned
  2      Unassigned
  3      Unassigned
  4      CHAN29      Parallel Channel
  5      Reserved
  6      Unassigned
  7      TOK1        Token-Ring 16/4
  8      Fixed Disk

 LAN Function Name ........... : SNAGATE
 Channel Adapter Name ........ : CHAN29

                To        To       LAN     Block  Maximum
 Subchannels  Channel     LAN    Adapter   Delay  Response
 -----------  --------  -------- --------  -----  --------
    60        TOCHN060  TOLAN060 TOK1       10      100

 Slot ........................ : 4
 Adapter Name ................ : CHAN29
 Adapter Type ................ : Parallel Channel
 Transfer Mode and Speed ..... : 4.5 MB Data Streaming
 SNA Management Services ..... : No

 Slot ........................ : 7
 Adapter Name ................ : TOK1
 Adapter Type ................ : Token-Ring 16/4
 Relative Adapter Number ..... : 0
 Node Address ................ : 400020201003
 Data Rate (Mbps) ............ : 4
 To Operator Facility ........ : No
 Combined Functional Addresses : 000000000000

 IEEE 802.2 (LLC)
   Response Timer (T1) ....... : 10  =  2000 ms
   Acknowledgment Timer (T2) . : 1   =    80 ms
   Inactivity Timer ....(Ti) . : 250 = 30000 ms
```

Figure 9. IBM 3172 ICP Configuration

4.2.4 Customization in VSE/VTAM

The following VTAM definitions in VSE VTAM are required to implement our network configuration:

- ***XCA Major Node.*** To define the 3172.
- ***Switched Major Node.*** To define the LAN resources that VTAM can address through the 3172.
- ***Logmode Table entry.*** To define the rules and protocols by which sessions between CICS/VSE and ezBRIDGE on OS/2 will be governed.
- ***VTAM APPL Major Node*** To define the VTAM facilities available to our host CICS/VSE system.

4.2.4.1 VTAM XCA Major Node

An External Communication Adapter (XCA) major node is required to enable VTAM to use a 3172. Figure 10 describes our 3172 XCA major node.

- Note that the ***CUADDR*** matches the virtual address which we have ADDed in our VSE/ESA IPL procedure in Figure 8 on page 21.

```
*************************************************************************
*        XCA MAJOR NODE                                                 *
*************************************************************************
BOEXCA   VBUILD TYPE=XCA
*
* DEFINITION FOR 3172
*
BOE3172  PORT  CUADDR=960,            PORT ADDRESS                    X
               MEDIUM=RING,           IBM TOKEN RING LAN              X
               ADAPNO=0,              RELATIVE ADAPTER NUMBER         X
               TIMER=60         (SEC) VTAM WAIT AFTER CHANNEL ACTIVATE
G3172    GROUP DIAL=YES               SWITCHED ATTACHMENT
L317201  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317201  PU    ISTATUS=ACTIVE
L317202  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317202  PU    ISTATUS=ACTIVE
L317203  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317203  PU    ISTATUS=ACTIVE
L317204  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317204  PU    ISTATUS=ACTIVE
L317205  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317205  PU    ISTATUS=ACTIVE
L317206  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317206  PU    ISTATUS=ACTIVE
L317207  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317207  PU    ISTATUS=ACTIVE
L317208  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317208  PU    ISTATUS=ACTIVE
L317209  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P317209  PU    ISTATUS=ACTIVE
L3172OA  LINE  ISTATUS=ACTIVE,CALL=INOUT,ANSWER=ON
P3172OA  PU    ISTATUS=ACTIVE
```

Figure 10. VSE/VTAM XCA Major Node for 3172

4.2.4.2 VSE/VTAM Switched Major Node

A Switched major node is created to define the physical and logical units to which VTAM may communicate through the 3172 XCA major node. Figure 11 on page 25 shows the definitions we used:

Note the following parameters:

- ***IDBLK*** is 05D. This is the IDBLK used for a PS/2 running Comms Manager/2.
- ***IDNUM*** is assigned locally and will be used later in our OS/2 CM/2 definitions.
- ***PATH*** The last 12 digits of the DIALNO operand contain the MACADDR of our primary PS/2 system.
- ***LOCADDR=0*** implies Independent Logical Unit.
- ***DLOGMODE=LU62PS*** is the name of the log mode entry which describes the session parameters for the APPC conversations between ezBRIDGE on VSE/ESA and ezBRIDGE on OS/2. See Figure 12 on page 26 for details of the log mode entry.

The last four LU definitions, IPFT2X7A, IPFT2X7B, IPFT2X7C, and IPFT2X7D in our example are used for 3270 sessions on the PS/2. These are not required for ezBRIDGE, but helped us to set up our environment.

```
*
*  3172 RELATED SWITCHED MAJOR NODE FOR PRIMARY ezBRIDGE PS/2
*
BOEXCASW       VBUILD TYPE=Switched,MAXGRP=20,MAXNO=20
*
*   PRIMARY PS/2 FOR ezBRIDGE
*
IPFCPX07 PU    ADDR=01,                                              X
               LANSW=YES,          LAN CAPABLE                       X
               IDBLK=05D,          IDENTIFICATION BLOCK
               IDNUM=E0007,        IDENTIFICATION NUMBER
               DISCNT=NO,          VTAM DOES NOT HANG UP             X
               ISTATUS=ACTIVE,PACING=1,VPACING=2,                    X
               PUTYPE=2,           SNA CLUSTER CONTROLLER            X
               MAXDATA=265,        MAX NUMBER OF BYTES HANDLED BY PU X
               MAXOUT=1,           MAX PIUS IN A BATCH               X
               MAXPATH=1,          MAX NUMBER OF DIAL PATHS          X
               SAPADDR=4           SERVICE ACCESS POINT ADDRESS
*
         PATH  DIALNO=0104400010101007
*
IPFXL070 LU    LOCADDR=0,DLOGMOD=LU62PS,                  (APPC LU)  X
               ISTATUS=ACTIVE,MODETAB=CICSIPMT
IPFT2X7A LU    LOCADDR=02,DLOGMOD=SP3273ES,                          X
               ISTATUS=ACTIVE,MODETAB=IESINCLM,                      X
               USSTAB=VTMUSSTR,MDLTAB=VTMMDL,MDLENT=VSELU2A,         X
               SSCPFM=USSSCS
IPFT2X7B LU    LOCADDR=03,DLOGMOD=SP3273ES,                          X
               ISTATUS=ACTIVE,MODETAB=IESINCLM,                      X
               USSTAB=VTMUSSTR,MDLTAB=VTMMDL,MDLENT=VSELU2A,         X
               SSCPFM=USSSCS
IPFT2X7C LU    LOCADDR=04,DLOGMOD=SP3273ES,                          X
               ISTATUS=ACTIVE,MODETAB=IESINCLM,                      X
               USSTAB=VTMUSSTR,MDLTAB=VTMMDL,MDLENT=VSELU2A,         X
               SSCPFM=USSSCS
IPFT2X7D LU    LOCADDR=05,DLOGMOD=SP3273ES,                          X
               ISTATUS=ACTIVE,MODETAB=IESINCLM,                      X
               USSTAB=VTMUSSTR,MDLTAB=VTMMDL,MDLENT=VSELU2A,         X
               SSCPFM=USSSCS
```

Figure 11. VSE/VTAM Switched Major Node for 3172

4.2.4.3 VSE/VTAM Logmode Table Entry

Log mode entries are required to define valid session parameters and the rules by which LUs will communicate.

Figure 12 on page 26 shows the logmode entry we used, LU62PS, as well as the standard logmode entry, SNASVCMG, which is distributed with VSE/VTAM for LU 6.2 sessions. This latter is used automatically by CICS/VSE for session control.

In practice, CICS and ezBRIDGE on OS/2 will negotiate PSERVIC values as part of the process of binding sessions. For an explanation of valid PSERVIC parameters for LU 6.2 sessions see the description of the BIND Request Unit in *SNA Network Product Formats, LY43-0081*. For an explanation of the other keywords on the MODEENT statement refer to *VTAM Resource Definition Reference, SC31-6438*.

```
CICSIPMT MODETAB
LU62PS   MODEENT  LOGMODE=LU62PS,                                    +
               TYPE=X'00',                                          +
               FMPROF=X'13',                                        +
               TSPROF=X'07',                                        +
               PRIPROT=X'B0',                                       +
               SECPROT=X'B0',                                       +
               COMPROT=X'50B1',                                     +
               RUSIZES=X'8989',                                     +
               SSNDPAC=X'00',                                       +
               SRCVPAC=X'00',                                       +
               PSNDPAC=X'00',                                       +
               PSERVIC=X'060200000000000000002F00'
SNASVCMG MODEENT  LOGMODE=SNASVCMG,                                 +
               FMPROF=X'13',                                        +
               TSPROF=X'07',                                        +
               PRIPROT=X'B0',                                       +
               SECPROT=X'B0',                                       +
               COMPROT=X'78A5',                                     +
               RUSIZES=X'8989',                                     +
               SSNDPAC=X'00',                                       +
               SRCVPAC=X'00',                                       +
               PSNDPAC=X'00',                                       +
               PSERVIC=X'060200000000000000122F00'
         MODEEND
         END
```

Figure 12. LOGMODE Entries

4.2.4.4 VTAM Application Major Node

A VTAM **Application Major Node** entry is required for the CICS/VSE system which controls ezBRIDGE on VSE/ESA. Since ezBRIDGE on VSE/ESA participates in the APPC communication to and from ezBRIDGE on OS/2 and ezBRIDGE on AIX the appropriate parameters have to be provided as shown in Figure 13 on page 27.

Note the following parameters:

- ***PARSESS=YES.*** Tells VTAM that this CICS can have parallel sessions with an Independent LU.
- ***EAS=4000.*** Tells VTAM that this CICS will have an estimated 4000 concurrent LU to LU sessions. This includes parallel sessions.
- ***APPC=NO.*** Tells VTAM that CICS does not use the VTAM APPCCMD macro instructions to converse via LU 6.2. CICS generates its own conversation streams.

```
BOEAPPL  VBUILD TYPE=APPL
DBDCCICS APPL  AUTH=(PASS,ACQ)
CICSSA22 APPL  AUTH=(PASS,ACQ,VPACE),PARSESS=YES,ACBNAME=CICSSA22,     C
               EAS=4000,MODETAB=CICSIPMT,APPC=NO,                      C
               SONSCIP=YES,VPACING=3
POWER    APPL  AUTH=(ACQ)                          DEFAULT RJE NAME
BOEPWNJE APPL  AUTH=(PASS,ACQ),VPACING=3,          ACB NAME FOR PNET    C
               MODETAB=NJEMODTB,                                       C
               DLOGMOD=NJEMOD
IESWAITT APPL  AUTH=(NOACQ)
```

Figure 13. Application Major Node

4.3 CICS/VSE Customization

The following resources have to be defined or updated in CICS/VSE to enable ezBRIDGE on VSE/ESA to communicate with ezBRIDGE on OS/2:

1. The appropriate CICS Tables (see 4.3.1, “CICS/VSE Resource Definition Using Macro”)
2. Connections to ezBRIDGE on OS/2 using SNA LU 6.2
3. Sessions to ezBRIDGE on OS/2 using SNA LU 6.2
4. Programs and transactions required for ezBRIDGE on VSE/ESA

There are two ways to define CICS resources in CICS/VSE:

- Macro Definition
- Resource Definition Online (RDO) using the CEDA transaction or the batch DFHCSDUP program

In our CICS/VSE, the Macro definition was used for SIT (System Initialization Table), File Control Table (FCT), Destination Control Table (DCT) and Terminal Control Program (TCP).

We used CEDA to define:

- Connections
- Sessions

We used the DFHCSDUP program to define:

- Programs
- Transactions

4.3.1 CICS/VSE Resource Definition Using Macro

The next four subsections describe the macro resource definitions necessary to define the ezBRIDGE on VSE/ESA product. The following resources are discussed:

- DFHSIT (System Initialization Table)
- prerequisites for DFHTCP (Terminal Control Program)
- DFHDCT (Destination Control Table).

- DFHFCT (File Control Table).

4.3.1.1 System Initialization Table (SIT)

The DFHSIT used in our implementation can be found in Figure 14 on page 29.

Critical parameters in the System Initialization Table are:

- ***APPLID.*** Represents this CICS. The name must match the:
  - ACBNAME operand of the APPL macro in the VSE VTAM Application Major Node
  - LU Name in the CM/2 Partner LU Definition
- ***ISC = YES.*** Includes the CICS group of programs for intercommunication. In our case, this is required to allow CICS/VSE to establish connections and sessions to our primary ezBRIDGE on OS/2 system which uses LU 6.2 protocol and DTP.
- ***JCT = SP.*** Journaling must be active for ezBRIDGE on VSE/ESA. We used the VSE/ESA distributed journal control table DFHJCTSP. If you do not already use journaling, see the skeleton in ICCF library 59 called SKJOURN for a sample implementation.

```
* $$ JOB JNM=DFHSITEZ,CLASS=A,DISP=D,NTFY=YES
* $$ LST CLASS=Q,DISP=H
// JOB DFHSITEZ ASSEMBLE
// LIBDEF *,CATALOG=PRD2.CONFIG
// OPTION CATAL,LIST
// EXEC ASSEMBLY
*************************************************
*                                               *
*   5686-028 (C) COPYRIGHT IBM CORP. 1984, 1990 *
*                                               *
*************************************************
         TITLE 'DFHSITEZ  --  FOR ezBRIDGE/OS2 TO VSE/ESA'
         PUNCH ' CATALOG DFHSITEZ.OBJ REP=YES'
         DFHSIT TYPE=CSECT,                                          *
               MXT=50,                 MAX NO. OF ALL CONCURRENT TASK *
                  .                                                  *
                  .                                                  *
               APPLID=CICSSA22,        CICS LUNAME   CICSSA22       *
               DCT=(EZ,COLD),          DESTID MQER AND CSMT         *
                  .                                                  *
                  .                                                  *
               FCT=EZ,                 INCLUDE APPLICATION FILES    *
                  .                                                  *
               JCT=SP,                 EZBRIDGE NEEDS JCT           *
                  .                                                  *
               ISC=YES,                INTERSYSTEM COMMUNICATION    *
                  .                                                  *
                  .                                                  *
                  .                                                  *
               ZCP=S$,                 ALL ACCESS METHODS           *
               DUMMY=DUMMY             TO END MACRO
         END   DFHSITBA
/*
// EXEC LNKEDT
/*
/&
* $$ EOJ
```

Figure 14. DFHSIT parameters

4.3.1.2 Terminal Control Program (DFHTCP)

The DFHSG PROGRAM=TCP macro instruction, used during CICS system generation and customization to create the terminal control program, should contain the following settings:

- ***ACCMETH=VTAM.*** To support VTAM
- ***CHNASSY=YES.*** To support SNA chain assembly
- ***VTAMDEV=LUTYPE6.*** To support LU6.2 Links

The supplied TCP (Suffix = S$) in CICS/VSE was generated with these parameters. You don't have to regenerate this program.

4.3.1.3 Destination Control Table (DFHDCT)

The job shown in Figure 15 is executed to assemble and catalog the destination control table.

MQER is an intrapartition transient data queue defined for use by ezBRIDGE on VSE/ESA.

The CSMT intrapartition transient data queue is also used by ezBRIDGE on VSE/ESA. The definition for CSMT is part of the standard DCT definition distributed with VSE/ESA.

```
* $$ JOB JNM=DFHDCTEZ,CLASS=A,DISP=D,NTFY=YES
* $$ LST CLASS=Q,DISP=H
// JOB DFHDCTEZ ASSEMBLE
// LIBDEF *,CATALOG=PRD2.CONFIG
// OPTION CATAL,LIST
// EXEC ASSEMBLY
*****************************************************
*   EZBRIDGE UPDATE: 03/09/94                       *
*   5686-028 (C) COPYRIGHT IBM CORP. 1984, 1990     *
*                                                   *
*****************************************************
         TITLE 'DFHDCTEZ -- INCLUDES EZ-BRIDGE SUPPORT'
         PUNCH ' CATALOG DFHDCTEZ.OBJ REP=YES'
         DFHDCT TYPE=INITIAL,SUFFIX=EZ
         EJECT
            .
            .
            .
MQER     DFHDCT TYPE=INTRA,                                  *
               RSL=PUBLIC,                                   *
               DESTID=MQER,                                  *
               DESTFAC=FILE,                                 *
               TRANSID=MQER,                                 *
               TRIGLEV=1
*   EZBRIDGE TRANSACT ALSO WILL USE CSMT
*   end  of ezBRIDGE Transact DCT entries
            .
            .
            .
         DFHDCT TYPE=FINAL
         END   DFHDCTBA
/*
// EXEC LNKEDT
/*
/&
* $$ EOJ
```

Figure 15. DFHDCT Entry for ezBRIDGE on VSE/ESA

4.3.1.4 File Control Table (DFHFCT)

The control files and message queues required by ezBRIDGE are mapped to VSE/VSAM files which are under the control of CICS/VSE. Figure 16 on page 31 shows the corresponding entries in the FCT to define these message queues and control files.

```
// JOB DFHFCTEZ ASSEMBLE
// LIBDEF *,CATALOG=PRD2.CONFIG
// OPTION CATAL,LIST
// EXEC ASSEMBLY
         TITLE 'DFHFCTEZ -- EZ BRIDGE ADDITIONS'
         PUNCH ' CATALOG DFHFCTEZ.OBJ REP=YES'
         DFHFCT TYPE=INITIAL,SUFFIX=EZ
                   .
                   .
*-------------------------------------------------------------------*
*  ezBRIDGE Transact Control Files                                  *
*-------------------------------------------------------------------*
* configuration file
MQFCNFG  DFHFCT TYPE=DATASET,DATASET=MQFCNFG,                        *
               ACCMETH=VSAM,                                         *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),              *
               LOG=YES,                                              *
               RSL=PUBLIC,                                           *
               BUFND=5,BUFNI=10,STRNO=20,     5 10 20                *
               RECFORM=(FIXED,BLOCKED)
* log queue
MQFLOG   DFHFCT TYPE=DATASET,DATASET=MQFLOG,                         *
               ACCMETH=VSAM,                                         *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),              *
               RSL=PUBLIC,                                           *
               LOG=YES,                                              *
               BUFND=10,BUFNI=10,STRNO=10,                           *
               RECFORM=(VARIABLE,BLOCKED)
* dead letter queue
MQFERR   DFHFCT TYPE=DATASET,DATASET=MQFERR,                         *
               ACCMETH=VSAM,                                         *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),              *
               RSL=PUBLIC,                                           *
               LOG=YES,                                              *
               BUFND=10,BUFNI=10,STRNO=10,                           *
               RECFORM=(VARIABLE,BLOCKED)
* monitor queue
MQFMON   DFHFCT TYPE=DATASET,DATASET=MQFMON,                         *
               ACCMETH=VSAM,                                         *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),              *
               RSL=PUBLIC,                                           *
               LOG=YES,                                              *
               BUFND=10,BUFNI=10,STRNO=10,                           *
               RECFORM=(VARIABLE,BLOCKED)
*-------------------------------------------------------------------*
*  ezBRIDGE Transact Message Queues                                 *
*-------------------------------------------------------------------*
* general local queue
LQVSE    DFHFCT TYPE=DATASET,DATASET=LQVSE,                          *
               ACCMETH=VSAM,                                         *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),              *
               LOG=YES,                                              *
               RSL=PUBLIC,                                           *
               BUFND=10,BUFNI=10,STRNO=10,                           *
               RECFORM=(VARIABLE,BLOCKED)
```

Figure 16 (Part 1 of 2). DFHFCT Entries for ezBRIDGE on VSE/ESA

```
* transmission queue to OS2S
QMOS2S   DFHFCT TYPE=DATASET,DATASET=QMOS2S,                          *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               LOG=YES,                                               *
               RSL=PUBLIC,                                            *
               BUFND=10,BUFNI=10,STRNO=10,                            *
               RECFORM=(VARIABLE,BLOCKED)
* local queue FROMOS2S
FROMOS2  DFHFCT TYPE=DATASET,DATASET=FROMOS2,                         *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               LOG=YES,                                               *
               RSL=PUBLIC,                                            *
               BUFND=10,BUFNI=10,STRNO=10,                            *
               RECFORM=(VARIABLE,BLOCKED)
* transmission queue to aix
QMAIX    DFHFCT TYPE=DATASET,DATASET=QMAIX,                           *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               LOG=YES,                                               *
               RSL=PUBLIC,                                            *
               BUFND=10,BUFNI=10,STRNO=10,                            *
               RECFORM=(VARIABLE,BLOCKED)
* local queue from aix
FROMAIX  DFHFCT TYPE=DATASET,DATASET=FROMAIX,                         *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               LOG=YES,                                               *
               RSL=PUBLIC,                                            *
               BUFND=10,BUFNI=10,STRNO=10,                            *
               RECFORM=(VARIABLE,BLOCKED)
         DFHFCT TYPE=FINAL
         END   DFHFCTBA
/*
// EXEC LNKEDT
/*
/&
```

Figure 16 (Part 2 of 2). DFHFCT Entries for ezBRIDGE on VSE/ESA

4.3.2 CICS/VSE Resource Definition Online (RDO)

We used the CEDA transaction to define the following resources to CICS:

- Connections
- Sessions

These resources are defined in the group **MQIOS2** in CICS/VSE. The following command shows the contents of the group CICSOS2:

- ***CEDA DISPLAY GROUP(MQIOS2)***

After defining the resources we used:

- ***CEDA INSTALL GROUP(MQIOS2)***
  To activate CICSOS2

- ***CEDA ADD GROUP(MQIOS2) LIST(VSELIST)***
  To make MQIOS2 a permanent entry in the list which is started when CICS starts.

Screen images of the CEDA sessions are given in this section, to enable you to quickly reproduce the definitions in your installation.

```
 EXPAND GROUP(MQIOS2)

  NAME     TYPE         GROUP
  OS2M     CONNECTION   MQIOS2
  OS2M     SESSIONS     MQIOS2
```

Figure 17. Group Definition of MQIOS2

4.3.2.1 Connection Definition

We defined the connection shown in Figure 18 in order to communicate to OS/2 workstation 1.

```
OBJECT CHARACTERISTICS
 CEDA  View
  Connection     : OS2M
  Group          : MQIOS2
 CONNECTION IDENTIFIERS
  Netname        : IPFXLO70
  INDsys         :
 REMOTE ATTRIBUTES
  REMOTESystem   :
  REMOTEName     :
 CONNECTION PROPERTIES
  ACcessmethod   : Vtam              Vtam | IRc | INdirect
  Protocol       : Appc              Appc | Lu61
  SInglesess     : No                No | Yes
  Datastream     : User              User | 3270 | SCs | STrfield | Lms
  RECordformat   : U                 U | Vb
 OPERATIONAL PROPERTIES
  AUtoconnect    : All               No | Yes | All
  INService      : Yes               Yes | No
 CONNECTION PROPERTIES
  ACcessmethod   : Vtam              Vtam | IRc | INdirect
  Protocol       : Appc              Appc | Lu61
  SInglesess     : No                No | Yes
  Datastream     : User              User | 3270 | SCs | STrfield | Lms
  RECordformat   : U                 U | Vb
 OPERATIONAL PROPERTIES
  AUtoconnect    : All               No | Yes | All
  INService      : Yes               Yes | No
 SECURITY
  SEcurityname   :
  ATtachsec      : Local             Local | Identify | Verify
  Bindpassword   :                   PASSWORD NOT SPECIFIED
```

Figure 18. Connection to Primary ezBRIDGE on OS/2 system

The key parameters are:

- ***CONNECTION.*** The name of the connection. It must match the:

  - CONNECTION name in CICS/VSE SESSIONS definition in Figure 19 on page 35

- ***GROUP.*** The name of the group to which this definition belongs
- ***NETNAME.*** The name of the independent LU. It must match the:
  - LU name in the CM/2 Optional SNA Features Local LU definition as in Figure 40 on page 59
  - Local LU Name in ezBRIDGE on OS/2′s channel definitions as in Figure 55 on page 72
- ***ACCESSMETHOD.*** Defines VTAM as the access method
- ***PROTOCOL.*** Defines APPC to be used for Distributed Transaction Processing programs
- ***SINGLESESS.*** Defines NO in order to use APPC parallel sessions

4.3.2.2 Session Definition

A session definition is used to define the logical links and the session characteristics between CICS/VSE and the independent LU (IPFXL070). Figure Figure 19 on page 35 shows the key session parameters for LU IPFXL070, our primary ezBRIDGE on OS/2 system on OS/2 workstation 1.

```
CEDA  View
 Sessions       : OS2M
 Group          : MQIOS2
SESSION IDENTIFIERS
 Connection     : OS2M
 SESSName       :
 NETnameq       :
 MOdename       : LU62PS
SESSION PROPERTIES
 Protocol       : Appc              Appc | Lu61
 MAximum        : 00008 , 00004     0-32767
 RECEIVEPfx     :
 RECEIVECount   : No                No | 1-999
 SENDPfx        :
 SENDCount      : No                No | 1-999
 SENDSize       : 00256             1-30720
 RECEIVESize    : 00256             1-30720
OPERATOR DEFAULTS
 OPERId         :
 OPERPriority   : 000               0-255
 OPERRsl        : 0
 OPERSecurity   : 1
 USERId         :
SESSION USAGES
 Transaction    :
 SESSPriority   : 100               0-255
OPERATIONAL PROPERTIES
 Autoconnect    : Yes               No | Yes | All
 INservice      :                   No | Yes
 Buildchain     : Yes               Yes | No
 USERArealen    : 000               0-255
 IOarealen      : 00000 , 00000     0-32767
 RELreq         : No                No | Yes
 Discreq        : No                No | Yes
 NEPclass       : 000               0-255
RECOVERY
 RECOvoption    : Sysdefault        Sysdefault | None
```

Figure 19. Session Definition for ezBRIDGE on OS/2 Connection

The key parameters are:

- ***SESSIONS.*** The name of the session.
- ***GROUP.*** The name of the group to which this definition belongs.
- ***CONNECTION.*** Defines the name of the connection associated with this session. It must match the:
  - CONNECTION name in CICS/VSE's CONNECTION definition as in Figure 18 on page 33
- ***MODENAME.*** The logmode entry name for the Independent LU named IPFXL070. It must match:
  - the MODEENT name in the VTAM MODETAB as in Figure 12 on page 26
  - an entry in CM/2's SNA Features Modes table as in Figure 43 on page 61
- ***PROTOCOL.*** APPC is required for ezBRIDGE
- ***MAXIMUM.*** This parameter has two values.
  The first value defines the maximum number of sessions supported. This MAXIMUM value specified here must match the:

– Mode Session Limit in CM/2′s optional SNA Features Modes as in Figure 43 on page 61

In our environment, a maximum of eight sessions was ample to support our message traffic. Check CICS/VSE shutdown statistics for session usage to refine this number.
The second value defines the maximum number of sessions which can be contention winners on the CICS/VSE end of the connection. The sum of this number and the maximum number of contention winners defined to CM/2 should equal the total for MAXIMUM sessions. In our case, specifying that four sessions should be contention winners at the host means we should specify that four sessions be contention winners at the ezBRIDGE on OS/2 system.

- ***SENDSIZE.*** The RU size for sending data. This value should match the:
  - Maximum RU Size in CM/2′s optional SNA Features Mode definition as in Figure 43 on page 61
  - RUSIZES in VTAM′s MODEENT definition in Figure 12 on page 26

    The send and receive RUSIZEs for the VTAM MODEENT must be specified in a special notation. The first half byte of each byte represents the mantissa, the second represents the exponent of the number 2. For example, a 4096 byte send length and a 640 byte receive length would be specified as RUSIZE=X′89A6′. Send being 8 times (2 to the power of 9) and receive being 10 times (2 to the power of 5). For further details, please refer to *VTAM Resource Definition Reference, SC31-6438*.
- ***RECEIVESIZE.*** The RU size for receiving data. This value should match the:
  - Maximum RU Size in CM/2′s optional SNA Features Mode definition as in Figure 43 on page 61
  - RUSIZES in VTAM′s MODEENT definition in Figure 12 on page 26
- ***AUTOCONNECT.*** YES allows CICS to establish a session automatically with the session partner IPFXL070 during CICS initialization
- ***BUILDCHAIN.*** Enables SNA Chain Assembly
- ***RELREQ.*** NO means that CICS/VSE is not to release the LU upon request by another VTAM application.
- ***DISCREQ.*** YES means that CICS/VSE will honor disconnect requests for a VTAM device, and issue a VTAM CLSDST macro to terminate the VTAM session with that LU.

4.3.2.3 Program and Transaction Definition

The program and transaction definitions for ezBRIDGE on VSE/ESA were installed using the three jobs listed in appendix A.3, “Define Programs and Transactions” on page 150.

The source PCT and PPT entries in the ezBRIDGE on VSE/ESA software library were assembled, and the offline RDO utility DFHCSDUP was used to migrate the tables into the CICS System Definition (CSD) file. The resulting CSD groups were **INSTALL**ed, then **ADD**ed to the CICS/VSE startup list named *′VSELIST′* using the following commands:

- ***CEDA INSTALL GROUP(EZPPT)***
  To activate EZPPT
- ***CEDA ADD GROUP(EZPPT) LIST(VSELIST)***

To make EZPPT a permanent entry in the list which is started when CICS starts.

- ***CEDA INSTALL GROUP(EZPCT)***
  To activate EZPCT
- ***CEDA ADD GROUP(EZPCT) LIST(VSELIST)***
  To make EZPCT a permanent entry in the list which is started when CICS starts.

4.4 ezBRIDGE on VSE/ESA Customization

This section provides the required definitions regarding ezBRIDGE queues and their management within VSE/ESA. The installation of the product itself is described in detail in *Messaging and Queuing Extensions for VSE/ESA, GG24-4296*.

ezBRIDGE lets you transfer messages **locally**, that is to queues managed by the **same** queue manager or to **remote** queues, which are under the control of another queue manager on the same or another system.

For the **local** case we need to define:

- A queue manager for this VSE/ESA system
- One or more local queues

If messages are to be sent to or received from other ezBRIDGE systems then the following items must be defined as well:

- One or more remote queues
- One or more transmission queues
- message channels for sending and receiving messages

Important Notes

1. During the definition of the queue manager at least the VSAM cluster for ezBRIDGE's control file ***MQFCNFG*** will be opened. This requires that the following actions must have been done:
   - Define a VSAM cluster for MQFCNFG using IDCAMS (refer to A.4, "VSAM Definitions for ezBRIDGE on VSE/ESA" on page 151 for the corresponding VSAM definitions).
   - Insert the corresponding '// DLBL' card for MQFCNFG into the standard labels or CICS job.
   - Define an entry for this file in the FCT table.
2. When defining a queue using the ***MQMT*** transaction the following items must be defined in advance:
   - For each queue definition a corresponding VSAM cluster must have been defined, unless one VSAM cluster will be used by more than one local queue. Sharing a VSAM cluster for two local queues or more might have an impact on the performance of MQI functions. Sharing a VSAM file amongst queues is **NOT** achieved via the SHARE MODE = Y parameter in the local queue definition, but just by using the same file name on more than one local queue definition. See Figure 23 on page 41 for a description of the SHARE MODE parameter.

     Refer to A.4, "VSAM Definitions for ezBRIDGE on VSE/ESA" on page 151 for the corresponding VSAM definitions.
   - For each VSAM file the corresponding '// DLBL card' has to be present either in the standard labels or in the job's JCL statements.
   - An appropriate entry has to be made in the FCT.
3. Before a remote queue can be defined the corresponding transmission queue must exist.

4.4.1 ezBRIDGE on VSE/ESA Customization for Local Use

A queue manager and at least one local queue have to be defined for the local case.

4.4.1.1 Defining the Queue Manager

Only **one** queue manager can run within a CICS/VSE system. It is defined via the 'Global System Definitions' option provided in MQMT's main menu. Figure 20 on page 39 shows our queue manager named *QMVSE* that we defined for our VSE/ESA host.

```
05/19/94            ezBRIDGE Transact for the MQSeries              CICSSA22
11:50:54                   Global System Definition                 CICS
MQFCNFG                 System QUEUE Manager Information            A001
 System Queue Manager. . . .: QMVSE
 Description line 1. . . . .: Q MGR ON VSE
 Description line 2. . . . .:
                         Channel Maximum Values
 Checkpointer Global timer .: 00000030
                         Queue System  Values
 Maximum number of MQCONN. .: 00000100      System  Wait Interval: 00000300
 Maximum Open Queues . . . .: 00000100      Allow internal DUMP  : Y
                         Queue Maximum Values
 Maximum Q Depth . . . . . .: 00100000      Maximum Global Locks.: 00000100
 Maximum message size. . . .: 00003353      Maximum Local Locks .: 00000100
 Maximum number of Opens . .: 00000100      Checkpoint Threshold : 1000
                         Global QUEUE /File Names
 Configuration File. . . . .: MQFCNFG
 LOG Queue name. . . . . . .: MQFLOG
 DEAD Letter Name. . . . . .: MQFERR
 Monitor Queue name. . . . .: MQFMON
RECORD DISPLAYED.

 PF2 = Main Config    PF3 = Quit     PF4/ENTER = Read       PF6 = Update

-------------------------------------------------------------------------
```

Figure 20. Define Queue Manager QMVSE

The key fields are:

- ***System Queue Manager.*** 'QMVSE' is the queue manager name on our VSE/ESA system.
- ***Maximum Q Depth.*** Maximum number of messages that all queues on this instance of ezBRIDGE can hold.
- ***Maximum message size.*** Maximum size of an individual message in bytes; this is determined by the CI-size of the corresponding VSAM file. For 4K CIs this number is 3353 bytes for user data, the rest is used for control information.
- ***Configuration file.*** The filename MQFCNFG will is provided by the MQMT dialog. FCT definition and '// DLBL card' file names have to match this name.
- ***LOG Queue name.*** Name of ezBRIDGE's queue for log information and error messages.
- ***DEAD Letter Name.*** Name of the file where message channels write messages that are received with wrong queue manager or queue names.
- ***Monitor Queue name.*** Name of the queue required when system monitoring is active.

  The filenames MQFLOG, MQFERR and MQFMON must match the names specified in the FCT and the corresponding '// DLBL' cards.
- All other parameters were left at their default values.

4.4.1.2 Defining a Local Queue

The local queue 'LQVSE' may be used for reading and writing messages, that is it may serve as local queue during IVT. It corresponds to the local queue named '*ANYQUEUE* mentioned for that purpose in the ezBRIDGE on VSE/ESA documentation. We decided to use a specific name on each platform.

To define a local queue a sequence of three screens is used as shown below.

```
05/19/94           ezBRIDGE Transact for the MQSeries              CICSSA22
08:38:25                Queue Main Options Screen                  CICS
MQFCNFG                                                            AOO1
                             SYSTEM IS ACTIVE

        Default Q Manager  : QM - QMVSE

                Object Type: L     L=Local Q, R=Remote Q, AQ=Alias Queue
                                                          AM=Alias Manager
                                                          AR=Alias Reply Q

                Object Name: LQVSE



                Like  Name :




ENTER NEEDED INFORMATION.

    PF2=Main Config   PF3=exit   PF4/ENTER=Read  PF5 = Add       PF6 = Update
                                                 PF9 = List      PF12= Delete
```

Figure 21. Define Local Queue LQVSE (1 of 3)

The key fields are:

- ***Object Type*** '***L***' defines a ***local*** queue.
- ***Object Name***' LQVSE' is the file name for this local queue.

```
05/19/94            ezBRIDGE Transact for the MQSeries              CICSSA22
08:42:06                Queue Definition  Record                    CICS
MQFCNFG           QM - QMVSE                                        A001

                      LOCAL QUEUE DEFINITION

 Object Name. . . . . . . . : LQVSE
 Description line 1 . . . . : LOCAL Q VSE
 Description line 2 . . . . :

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No









 RECORD BEING ADDED - PRESS ADD KEY AGAIN.

     PF2 = Options   PF3 = Quit    PF4/ENTER = Read   PF5 = Add    PF6 = Update
                     PF9 = List    PF10= Extended                  PF12= Delete
--------------------------------------------------------------------------------
```

Figure 22. Define Local Queue LQVSE (2 of 3)

The key fields are:

- ***Put enabled.*** 'Y' allows for writing to this queue.
- ***Get enabled.*** 'Y' allows for reading from this queue.

```
05/19/94           ezBRIDGE Transact for the MQSeries               CICSSA22
09:01:51             Local QUEUE Extended Definition                CICS
MQFCNFG           QM - QMVSE                                        A001
Object name. . . . . . . . : LQVSE
                         Local Queue Information
Usage mode . . . . . . . . : N     N=Normal, T=Transmission
Share mode . . . . . . . . : N     Y=Yes, N=No
Physical File name . . . . : LQVSE      MQSERIES.LQVSE
                            Maximum Values
Maximum Q Depth. . . . . . : 00100000     Global Lock entries . : 00000100
Maximum message length . . : 00003353     Local Lock entries. . : 00000100
Maximum number of Opens . .  00000100     Checkpoint threshold  : 1000

                         Trigger Information
Trigger enable . . . . . . : N     Y=yes, N=No
Trigger Type   . . . . . . :       F=First, E=every
Maximum Trigger starts . . : 0001
Trans ID   :           Term ID : A004     SYSID      :
Program ID :                              Remote CID :

RECORD BEING ADDED - PRESS ADD KEY AGAIN.

    PF2 = Options   PF3 = Quit     PF4/ENTER= Read    PF5 = Add  PF6 = Update
                    PF9 = List     PF10 = Queue                PF12 = Deletes
--------------------------------------------------------------------------------
```

Figure 23. Define Local Queue LQVSE (3 of 3)

The key fields are:

- ***Usage mode.*** ′N′ defines a ″normal″ local queue for reading (MQGET) and writing (MQPUT) messages in contrast to ′T′ for transmission queues.
- ***Share mode.*** ′YES′ provides for multiple concurrent access to that queue, that is more than one application can read from and write to it.
- ***Term id.*** Used for debugging. We specified our terminal id.
- All other parameters were left at their default values.

Having defined the queue manager and one local queue we can run an Installation Verification Test (IVT) to check whether a message can be written to and read from a local queue. The system supplied transaction *TST2* can be used to write to *LQVSE*.

4.4.2 ezBRIDGE on VSE/ESA to ezBRIDGE on OS/2 Configuration

In order to exchange messages with OS/2 workstation 1 (OS2S) and 2 (OS22) as shown in Figure 3 on page 15 and Figure 4 on page 17 we defined:

1. an additional local queue named **FROMOS2**
2. a transmission queue named **QMOS2S**
3. two remote queues named **TOOS2** and **TOOS22**
4. one message channel for sending messages named **CVSEOS2**
5. one message channel for receiving messages named **COS2VSE**

4.4.2.1 Defining Additional Local Queue FROMOS2

An additional local queue named FROMOS2 was defined for the messages coming from OS2S. The queue definitions correspond to the ones shown for LQVSE from Figure 21 on page 40 to Figure 23 on page 41. The only difference are the names used during the definition, that is ′LQVSE′ has to be replaced by ′FROMOS2′.

4.4.2.2 Defining a Transmission Queue

The queue QMOS2S can be considered as an ′output queue′ holding messages for the queue manager QMOS2S. We used the same name for both, the transmission queue on VSE/ESA and the queue manager on OS/2 which is the destination queue. A transmission queue is a local queue holding messages that are to be forwarded to a remote queue manager.

Transmission queues are defined in the same way as local queues, that is via a sequence of three screens as shown below. While the first two screens are the same as for local queue definition, the actual transmission queue definition is done in the third screen (see Figure 26 on page 44).

```
05/19/94          ezBRIDGE Transact for the MQSeries              CICSSA22
09:14:02               Queue Main Options Screen                  CICS
MQFCNFG                                                           A001
                            SYSTEM IS ACTIVE

        Default Q Manager  : QM - QMVSE

                Object Type: L     L=Local Q, R=Remote Q, AQ=Alias Queue
                                                          AM=Alias Manager
                                                          AR=Alias Reply Q

                Object Name: QMOS2S



                Like  Name :




 ENTER NEEDED INFORMATION.

     PF2=Main Config   PF3=exit   PF4/ENTER=Read  PF5 = Add       PF6 = Update
                                                  PF9 = List      PF12= Delete
-------------------------------------------------------------------------------
```

Figure 24. Define Transmission Queue QMOS2S (1 of 3)

```
05/19/94           ezBRIDGE Transact for the MQSeries              CICSSA22
09:14:37               Queue Definition  Record                   CICS
MQFCNFG          QM - QMVSE                                       A001

                      LOCAL QUEUE DEFINITION

 Object Name. . . . . . . . : QMOS2S
 Description line 1 . . . . : XMIT q to OS2S, fed
 Description line 2 . . . . : via rem q TOOS2

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No









 RECORD BEING ADDED - PRESS ADD KEY AGAIN.

     PF2 = Options   PF3 = Quit    PF4/ENTER = Read   PF5 = Add    PF6 = Update
                     PF9 = List    PF10= Extended                  PF12= Delete
-------------------------------------------------------------------------------
```

Figure 25. Define Transmission Queue QMOS2S (2 of 3)

```
05/19/94           ezBRIDGE Transact for the MQSeries                CICSSA22
09:14:57             Local QUEUE Extended Definition                 CICS
MQFCNFG           QM - QMVSE                                         A001
Object name. . . . . . . . : QMOS2S
                          Local Queue Information
Usage mode . . . . . . . . : T     N=Normal, T=Transmission
Share mode . . . . . . . . : N     Y=Yes, N=No
Physical File name . . . . : QMOS2S     MQSERIES.QMOS2S
                             Maximum Values
Maximum Q Depth. . . . . . : 00100000     Global Lock entries . : 00000100
Maximum message length . . : 00001024     Local Lock entries. . : 00000100
Maximum number of Opens . .  00000100     Checkpoint threshold  : 1000

                          Trigger Information
Trigger enable . . . . . . : Y     Y=yes, N=No
Trigger Type   . . . . . . : F     F=First, E=every
Maximum Trigger starts . . : 0001
Trans ID   :           Term ID :          SYSID      :
Program ID : MQPSEND                      Remote CID : CVSEOS2

RECORD BEING ADDED - PRESS ADD KEY AGAIN.

    PF2 = Options   PF3 = Quit     PF4/ENTER= Read    PF5 = Add  PF6 = Update
                    PF9 = List     PF10 = Queue                PF12 = Deletes
-------------------------------------------------------------------------------
```

Figure 26. Define Transmission Queue QMOS2S (3 of 3)

The key fields are:

- ***Object name.*** QMOS2S is the name of this transmission queue.
- ***Usage mode.*** 'T' for transmission queue. This determines how this queue is used. Messages are put into this queue by writing to the associated remote queue which is defined in the following step. Messages are read from this queue by a message channel agent to be defined in a corresponding message channel definition later on.
- ***Physical file name.*** QMOS2S matches the file name specified in the CICS FCT and corresponds to the VSAM file id MQSERIES.QMOS2S.
- ***Max message length.*** Must not exceed the limit defined in our queue manager (3353); we specified 1024 bytes. Exchange of messages between queues of different lengths is possible, the communicating MCA's 'negotiate' on an acceptable message size for both sides.
- ***Trigger enable.*** 'Y' starts the program defined below.
- ***Trigger type.*** 'F' for 'first' means that the triggering mechanism starts the message channel agent program when the first message is written to the transmission queue. 'E' for 'every' will start the program every time a message shows up at that queue.
- ***Maximum trigger starts.*** '1' means every single message should be sent out. A number greater than 1 means messages are kept in the queue until this number of messages is reached, that is messages are sent as a 'bunch'.
- ***Program ID.*** 'MQPSEND' is the program name for the MCA responsible for sending messages.
- ***Remote CID.*** 'CVSEOS2' is the name of the remote system's message channel definition (refer to Figure 45 on page 63).

4.4.2.3 Defining Remote Queues

As shown in Figure 4 on page 17 we defined two remote queues named TOOS2 and TOOS22. Both remote queues use transmission queue QMOS2S for sending messages to either workstation 1 (OS2S) or 2 (OS22).

1. Remote queue TOOS2 to queue manager QMOS2S on OS/2 workstation 1 (OS2S):

```
05/19/94           ezBRIDGE Transact for the MQSeries              CICSSA22
09:54:52                Queue Main Options Screen                  CICS
MQFCNFG                                                            AOO1
                             SYSTEM IS ACTIVE

        Default Q Manager  : QM - QMVSE

                Object Type: R     L=Local Q, R=Remote Q, AQ=Alias Queue
                                                          AM=Alias Manager
                                                          AR=Alias Reply Q

                Object Name: TOOS2



                Like  Name :



ENTER NEEDED INFORMATION.

    PF2=Main Config   PF3=exit   PF4/ENTER=Read  PF5 = Add       PF6 = Update
                                                 PF9 = List      PF12= Delete
-------------------------------------------------------------------------------
```

Figure 27. Define Remote Queue TOOS2 (1 of 2)

The key fields are:

- ***Object type.*** 'R' defines a queue.
- ***Object name.*** 'TOOS2' is the name of the remote queue.

```
05/19/94            ezBRIDGE Transact for the MQSeries              CICSSA22
09:55:29                Queue Definition  Record                    CICS
MQFCNFG           QM - QMVSE                                        A001

                       REMOTE QUEUE DEFINITION

 Object Name. . . . . . . . : TOOS2
 Description line 1 . . . . : REMOTE QUEUE TO OS2
 Description line 2 . . . . : USES QMOS2S

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No

 Remote QUEUE name . . . . .  FROMVSE
 Remote QM    name . . . . .  QMOS2S

 Transmission name . . . . .  QMOS2S



 RECORD BEING ADDED - PRESS ADD KEY AGAIN.

    PF2 = Options   PF3 = Quit    PF4/ENTER = Read   PF5 = Add    PF6 = Update
                    PF9 = List                                    PF12= Delete
-------------------------------------------------------------------------------
```

Figure 28. Define Remote Queue TOOS2 (2 of 2)

The key fields are:

- ***Remote queue name.*** 'FROMVSE' is the name of the local queue on the remote system (OS2S) to which messages are sent.
- ***Remote QM name.*** 'QMOS2S' is the name of the queue manager on the remote system OS2S.
- ***Transmission name.*** 'QMOS2S' is the name of the transmission queue defined in the previous step.

2. Remote queue TOOS22 to queue manager QMOS22 on OS/2 workstation 2 (OS22):

The definition of the second remote queue TOOS22 corresponds to the one made for TOOS2, with the exception of the queue name on the first screen.

On the second screen some care is required because we use OS/2 workstation 1 (OS2S) as 'MQI-Router' for OS/2 workstation 2 (OS22) as shown in Figure 4 on page 17.

```
05/19/94            ezBRIDGE Transact for the MQSeries              CICSSA22
09:55:29                Queue Definition  Record                    CICS
MQFCNFG           QM - QMVSE                                        A001

                       REMOTE QUEUE DEFINITION

 Object Name. . . . . . . . : TOOS22
 Description line 1 . . . . : REM Q TO OS22 via XMIT Q QMOS2S
 Description line 2 . . . . : to address Q MANAGER QMOS22

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No

 Remote QUEUE name . . . . .  FROMOS2S
 Remote QM    name . . . . .  QMOS22

 Transmission name . . . . .  QMOS22



 RECORD BEING ADDED - PRESS ADD KEY AGAIN.

     PF2 = Options   PF3 = Quit    PF4/ENTER = Read   PF5 = Add    PF6 = Update
                     PF9 = List                                    PF12= Delete
-------------------------------------------------------------------------------
```

Figure 29. Define Remote Queue TOOS22 (2 of 2)

The key fields are:

- ***Remote queue name.*** 'FROMOS2S' is the name of the local queue on the remote system (OS22) to which messages are sent.
- ***Remote QM name.*** 'QMOS22' is the name of the queue manager on the remote system OS22.
- ***Transmission name.*** 'QMOS22' is the name of the corresponding transmission queue on OS2S.

4.4.2.4 Defining Message Channels

Corresponding to our MQI environment illustrated in Figure 4 on page 17 we defined two message channels on VSE/ESA to exchange messages with OS2S, one for sending (CVSEOS2) and one for receiving (COS2VSE).

For an overview of the naming conventions used for message channel and message channel agent definitions refer to C.5, "Message Channel and MCA Summary" on page 163.

Choice number 3 of MQMT's main menu lets you create, modify or delete message channels.

1. Sender Channel CVSEOS2.

```
05/19/94          ezBRIDGE Transact for the MQSeries                CICSSA22
10:13:53                    Channel Record          DISPLAY         CICS
            Last check point               Last update              A001
MSN           Time           Interv        Create date

 Channel name  : CVSEOS2                    Channel type    S S=Send/R=Recv
 Protocol    L62    L62/BSC/LAN/X25          Format   MCP    MLP/MEP/MCP

        Allocation retries               Get retries
    Number of retries: 00000003       Number of retries:  00000003
    Delay time-fast :  00000002       Delay time      :   00000002
    Delay time-slow :  00000005

  Max messages per batch : 000001         Max transmission size  :  001200
  Message sequence wrap  : 001000         Max message size       :  001024

   Mess seq req(y/n): Y     Convers cap (y/n): N      Split mess(y/n):   N
             Connection ID: OS2M          Remote task ID: TPVS
 Transmit queue name        QMOS2S

 Checkpoint values:       Frequency   0020     Time span   0010
 Enable(Y/N) Y      Dead letter store(Y/N) Y
ENTER CHANNEL NAME.
PF2 =Menu  PF3 =Quit  PF4 =Read  PF5 =Add  PF6 =Update  PF9 =LIST  PF12 =Delete
--------------------------------------------------------------------------------
```

Figure 30. Define Sender Channel CVSEOS2

The key fields are:

- ***Channel type.*** 'S' defines a message channel for sending.
- ***Protocol.*** 'L62' defines an SNA LU 6.2 connection allowing APPC.
- ***Format.*** 'MCP' is the only format currently supported.
- ***Allocation retries.*** See next field for description.
- ***Get retries.*** These numbers refer to exceptional queue situations (queues can't be allocated or are depleted). We used the values shown in the screen above.
- ***Message sequence wrap.*** After 1000 messages the message sequence number will start over from 1 again.
- ***Max transmission size.*** Maximum number of bytes per transmission.
- ***Max message size.*** Maximum number of bytes per message.
- ***Mess seq req.*** 'Y' for yes will check for matching message sequence numbers on the sender and receiver side.
- ***Convers cap.*** Only 'N' for NO is allowed. No support for data conversion from ASCII to EBCDIC and vice versa is provided by ezBRIDGE.
- ***Connection ID.*** OS2M is the name of the connection definition in CICS, made via RDO (refer to Figure 19 on page 35).
- ***Remote task ID.*** 'TPVS' is the name of the CM/2 TP name definition that will be used on OS2S to start the receiving message channel agent (refer to Figure 45 on page 63).

- ***Transmit queue name.*** 'QMOS2S' is the transmission queue for sending messages to OS2S (refer to 4.4.2.2, "Defining a Transmission Queue" on page 42).
- ***Dead Letter store.*** 'Y' to allow storing of wrongly 'addressed' messages into the dead letter queue. Existence of a dead letter queue is recommended and should be defined in the same way as a local queue.

2. Receiver Channel COS2VSE.

```
05/19/94          ezBRIDGE Transact for the MQSeries                CICSSA22
10:15:45                    Channel Record          DISPLAY         CICS
            Last check point               Last update              A001
MSN           Time 08:27:30  Interv         Create date

 Channel name  : COS2VSE                     Channel type    R S=Send/R=Recv
 Protocol    L62    L62/BSC/LAN/X25          Format   MCP    MLP/MEP/MCP

        Allocation retries               Get retries
    Number of retries: 00000003       Number of retries:  00000003
    Delay time-fast :  00000002       Delay time      :   00000002
    Delay time-slow :  00000005

  Max messages per batch : 000001         Max transmission size  :  001200
  Message sequence wrap  : 001000         Max message size       :  001024

   Mess seq req(y/n): Y     Convers cap (y/n): N      Split mess(y/n):   N
             Connection ID:                Remote task ID:
 Transmit queue name

 Checkpoint values:       Frequency   0020     Time span   0010
 Enable(Y/N) Y      Dead letter store(Y/N) Y
ENTER CHANNEL NAME.
PF2 =Menu  PF3 =Quit  PF4 =Read  PF5 =Add  PF6 =Update  PF9 =LIST  PF12 =Delete
--------------------------------------------------------------------------------
```

Figure 31. Define Receiver Channel COS2VSE

Note: Most of the values are the same for both sender and receiver channels. Note, however the following differences:

- ***Channel name.*** 'COS2VSE' for the receiver channel.
- ***Channel type.*** 'R' defines a receiver channel.
- ***Connection ID.*** Leave blank. This name will be transmitted from the sender message channel agent.
- ***Remote task ID.*** Leave blank. For a receiving channel definition this name is not needed.
- ***Transmit queue name.*** Leave blank. A receiver will not use a transmission queue.

4.4.3 ezBRIDGE on VSE/ESA to ezBRIDGE on AIX Configuration

To provide for MQI communication between VSE/ESA and AIX the same concepts apply as for message queuing to OS/2. The definitions for the corresponding MQI scenario illustrated in Figure 81 on page 102 include:

1. On VSE/ESA:
   a. One local queue named FROMAIX.

b. One remote queue named TOAIX.

c. One transmission queue named QMAIX.

d. Two message channels for sending and receiving, named CVSEAIX and CAIXVSE.

2. On AIX:

a. One local queue named FROMVSE.

b. One remote queue named TOVSE.

c. One transmission queue named QMVSE.

d. Two message channels for sending and receiving, named CAIXVSE and CVSEAIX.

With the exception of their respective names the queue and message channel definitions in ezBRIDGE on VSE/ESA correspond exactly to the ones we made for exchanging messages with OS/2 workstations described in 4.4.2, “ezBRIDGE on VSE/ESA to ezBRIDGE on OS/2 Configuration” on page 42.

Chapter 5. ezBRIDGE on OS/2 Implementation

The following steps are required to implement ezBRIDGE on OS/2:

1. Install and customize LAPS
2. Customize CM/2
3. Install and customize ezBRIDGE on OS/2

All definitions and parameter settings made within these steps refer to our MQI test environment illustrated in Figure 3 on page 15 and Figure 4 on page 17. You are recommended to refer to these diagrams while reading this chapter.

5.1 LAPS Installation and Customization

Before ezBRIDGE on OS/2 can communicate over the Token-Ring to any other ezBRIDGE system, the LAPS program must be installed and customized. This is the program that physically manages the PS/2 LAN adapter.

To install LAPS from the product diskette, do the following:

1. Put the LAPS diskette into your diskette drive (for example A)
2. At the OS/2 prompt, type: **A:laps** and press the Enter key.

The window in Figure 32 on page 52 lists the features which we selected for the OS/2 workstation 1 (OS2S).

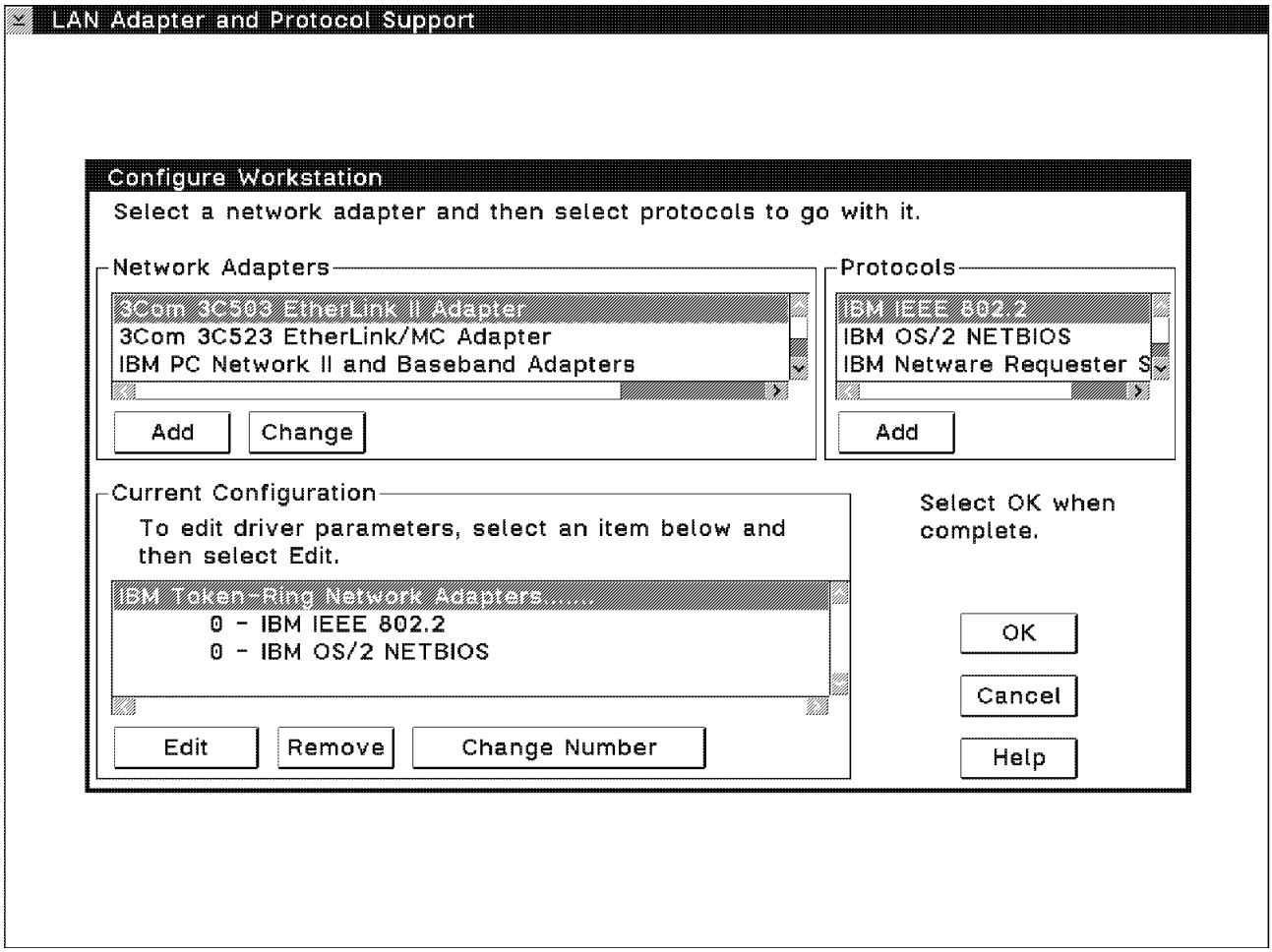

Figure 32. LAPS Configuration Menu for OS/2 Workstation 1 (OS2S)

In this panel you have to select the Network Adapter (refer to the PS/2 hardware setup) and then select one, or more, protocols to go with it. For our installation we selected the IBM Token-Ring Network Adapters with two protocols, IBM IEEE 802.2 and IBM OS/2 NetBIOS. These protocols allow us to communicate with other OS/2 systems, that is workstation 2 (OS22) over the Token-Ring. After the selection is made (clicking the item with mouse pointer and ″pushing″ the Add button), you must edit the driver parameter. The following window will be displayed:

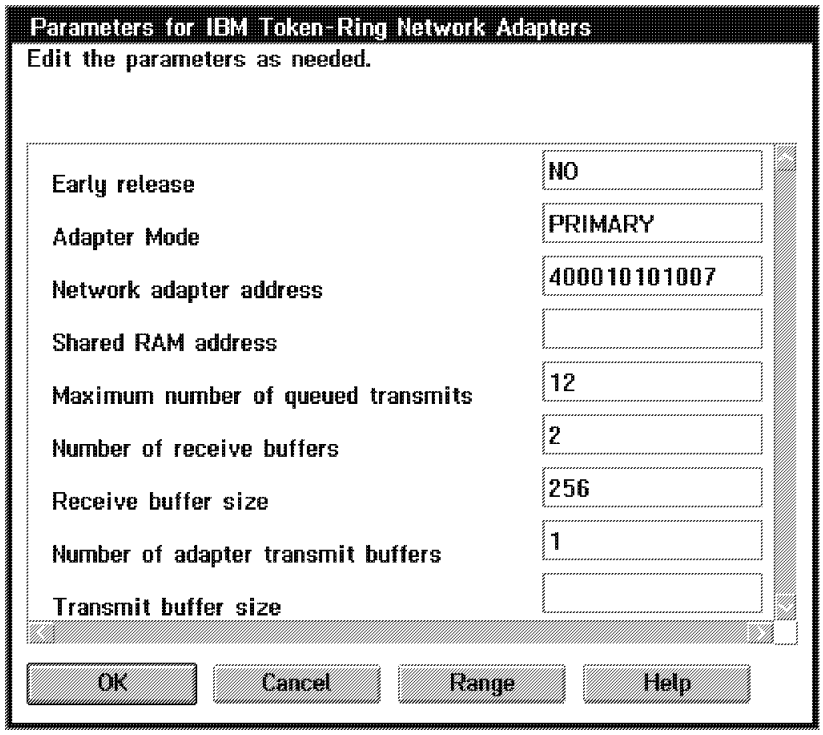

Figure 33. LAPS Token-Ring Adapter Customization

The only requested parameter to set up is the **Network Adapter Address**, for the others, the default values are good. This field specifies the locally administered address of the PS/2 Token-Ring adapter and **must match** the last six bytes of the **PATH** macro's **DIALNO** parameter in the VTAM Switched major node described in Figure 11 on page 25.

- 400010101007 is the Token-Ring MACADDRess assigned to our PS/2 named OS2S running ezBRIDGE on OS/2 (workstation 1 or primary).

After you have saved all your responses, re-boot the PS/2.

If you wish to change the definitions of LAPS again, you have to invoke LAPS.EXE which can be found in the C:IBMCOM directory.

5.2 CM/2 Customization for ezBRIDGE on OS/2

This section contains guidance information about the CM/2 definitions required if ezBRIDGE on OS/2 wants to communicate to ezBRIDGE systems on other platforms.

For the SNA LU 6.2 (APPC) connections required in our MQI environment we made the following definitions in the CM/2 configuration files:

1. One **DLC Adapter Parameters** definition, which specifies Data Link Control characteristics of the token-ring adapter.
2. One **Local node characteristics** definition, which specifies characteristics that are common to all APPC users on this workstation.
3. SNA **connection** definitions to VSE/ESA and OS/2 workstation 2 (OS22).
4. The necessary optional SNA features for SNA LU 6.2 communication to the VSE/ESA host and to workstation OS22 which in turn consists of:

a. A **Local LU** definition for OS2S and OS22.

b. A **Partner Logical Unit** definition for each remote ezBRIDGE system to which this workstation wants to communicate.

c. One or more **MODE** definitions to specify sets of session properties that are used in binding APPC sessions.

d. A **Transaction program** definition for every local transaction invoked when messages are sent to or received from remote systems.

With regard to data conversion which may be required when exchanging messages with other operating system platforms, CM/2 provides a sample conversion table, ***ACSGTAB.DAT***. This table represents the CM/2 default for EBCDIC to ASCII conversion. If ACSGTAB.DAT is inappropriate, you may provide and define your own table by specifying its name in the name field of the ***Change Workstation Information*** panel (CM/2 Configuration definitions -> Options -> Change Workstation Information).

The following panels show the definitions we made to enable our OS/2 workstation for SNA LU 6.2 (APPC) communication over a token-ring.

5.2.1 Configure DLC Token-Ring

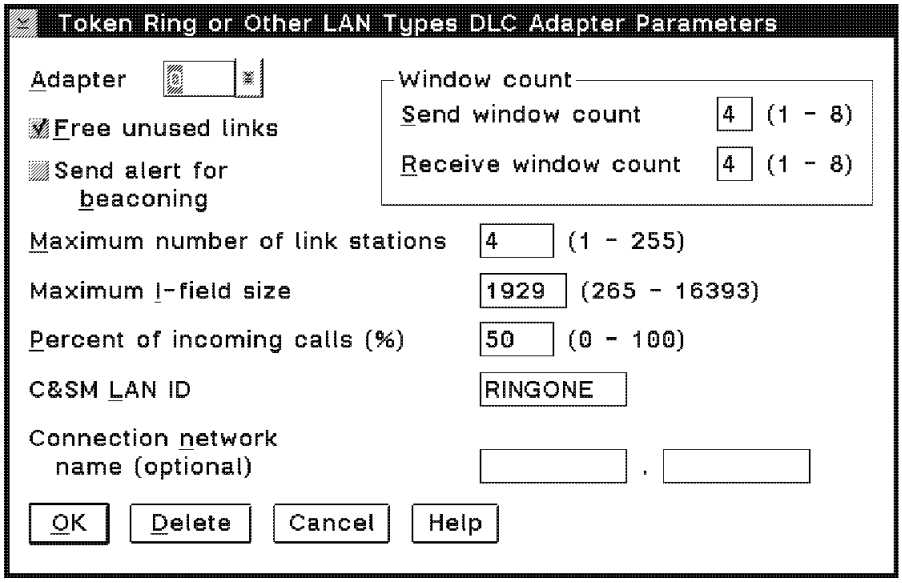

Figure 34. Token-Ring DLC Adapter Parameters

Most of the defaults are good, just provide a name for ***C&SM LAN ID*** if the network is to be identified by a LAN management tool.

5.2.2 Defining Local Node Characteristics

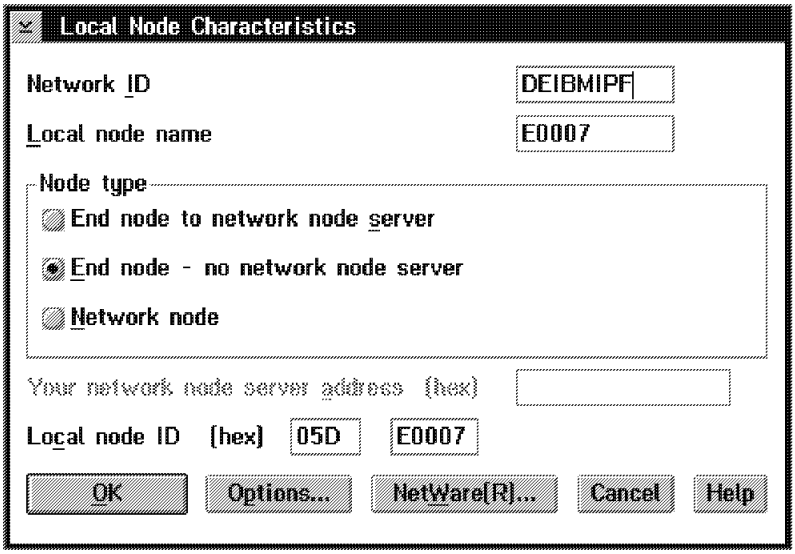

Figure 35. Local Node Characteristics

The key fields are:

- ***Network ID.*** DEIBMIPF is the name of the network to which this PU belongs. This must match the NETID parameter of the VTAM start list. See A.1, “VTAM Start List” on page 149.
- ***Node type.*** Our PS/2 is an end node accessed by the host through a 3172 gateway. Our 3172 gateway is **not** configured as a network node server.
- ***Local Node ID.*** The values 05D E0007 are used for the XID exchange. They match IDBLK and IDNUM values specified on our PU definition statement in the VTAM Switched Network major node as in Figure 11 on page 25.

Clicking the ’Options’ button brings us to the **Local Node Options** window shown in Figure 36 below.

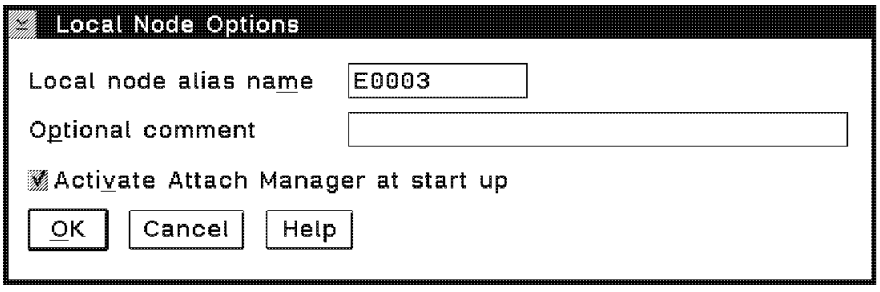

Figure 36. Local Node Options Alias Name

The key field is:

- **Activate Attach Manager**. This is a feature of APPC which manages incoming requests to begin a communication session. In our case, the Attach Manager is started automatically when the Communications Manager is started.

5.2.3 Define SNA Host Connections

5.2.3.1 To VSE/ESA Host

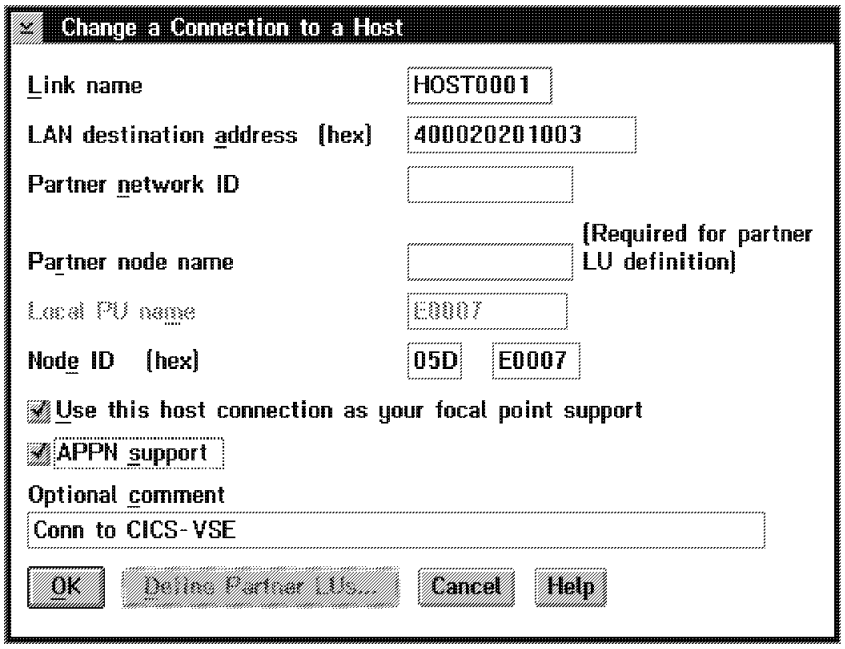

Figure 37. Connection Definition Panel to Host

The key fields are:

- ***Link Name.*** HOST0001 is the link name of the connection to the host.
- ***LAN Destination address.*** 400020201003 is the LAN MACADDRess of the 3172 gateway to our host as specified in the 3172 ICP configuration in Figure 9 on page 22.
- ***Node ID.*** 05D E0007 are the IDBLK and IDNUM parameters as specified in the VTAM switched major node as in Figure 11 on page 25.

5.2.3.2 To OS/2 Workstation 2 (OS22)

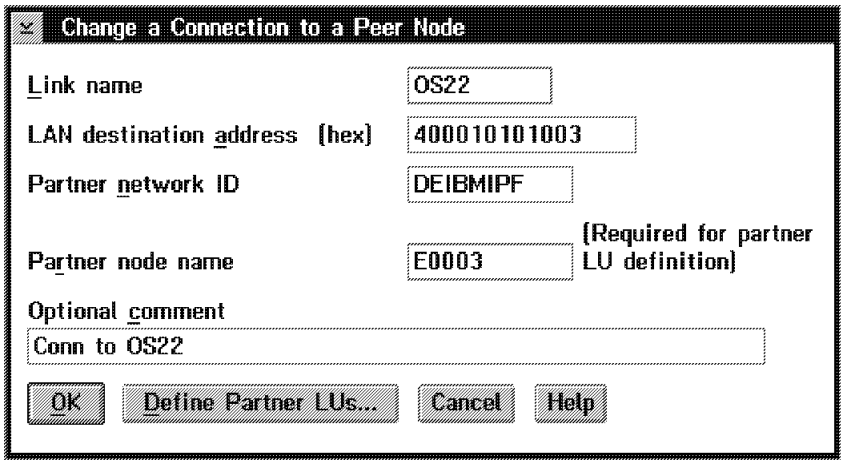

Figure 38. Connection Definition Panel to OS22

The key fields are:

- ***Link Name.*** OS22 is the link name of the connection to OS/2 workstation 2 (OS22).
- ***LAN Destination address.*** 400010101003 is the LAN MACADDRess of OS22's token-ring adapter.
- ***Partner node name.*** E0003 is the node name of OS22.

5.2.4 Defining Optional SNA Features

From the list shown in panel in Figure 39 we created the following SNA features in CM/2 of OS2S to provide for our APPC connections:

1. Local LUs
2. Partner LUs
3. Modes
4. Transaction program definitions

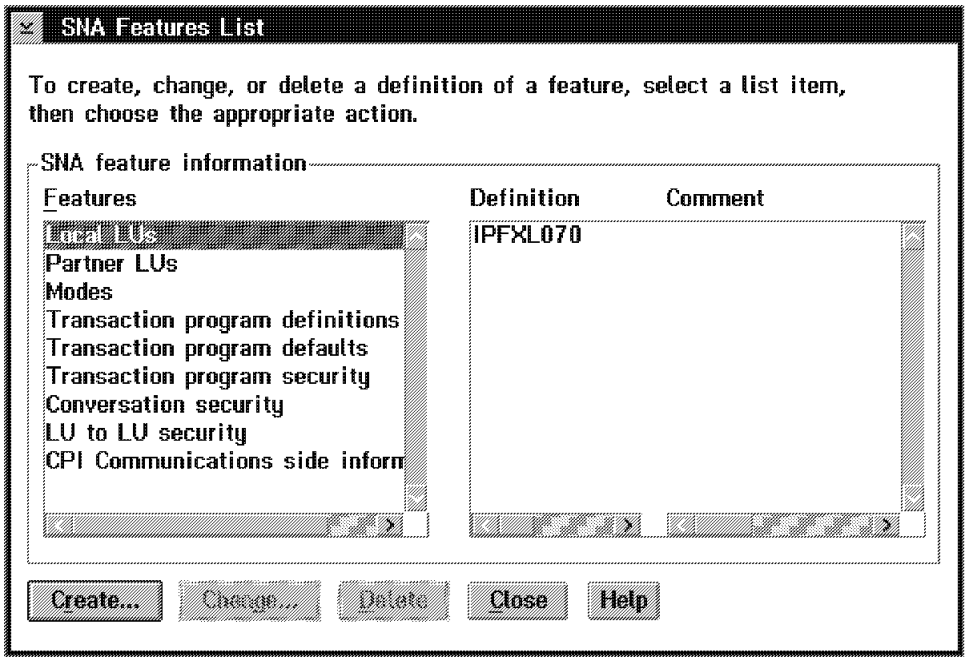

Figure 39. List of Optional SNA Features

5.2.4.1 Local LU Definition for OS2S

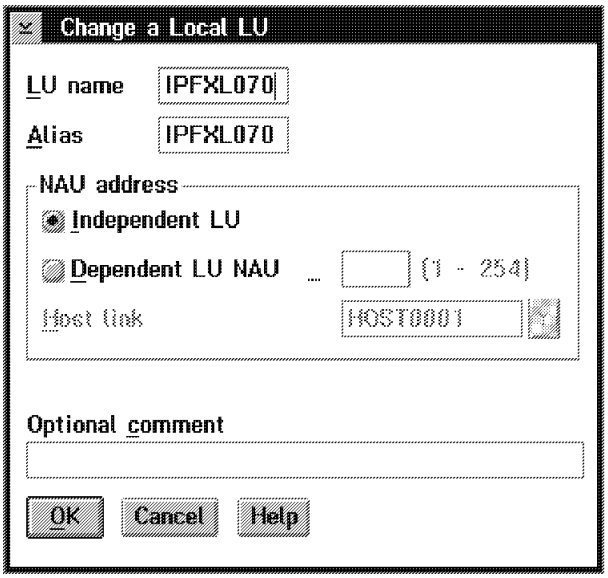

Figure 40. Local LU Definition Panel

The key fields are:

- ***LU Name.*** IPFXL070 is the name by which the Logical Unit is known throughout the SNA network. It matches the definition in the VSE/VTAM Switched network major node in Figure 11 on page 25.
- ***Alias.*** IPFXL070 is the alias name by which transaction programs may refer to this LU.
- ***NAU address.*** IPFXL070 is an independent LU.

5.2.4.2 Defining the Partner LUs for Workstation 1 (OS2S)

There are two partners to which OS2S wants to communicate:

1. ezBRIDGE on VSE/ESA running under CICS/VSE, represented by the LU named *'CICSSA22'*, see Figure 41 and
2. ezBRIDGE on OS/2 running on workstation 2 (OS22), represented by the LU named *'IPFXL030'*, see Figure 42.

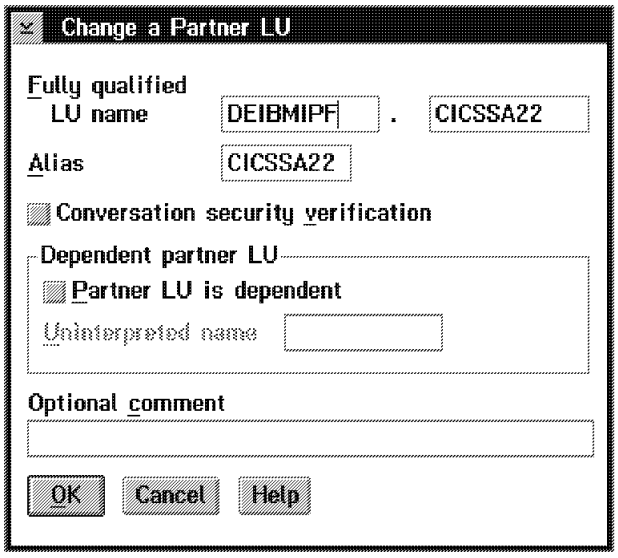

Figure 41. Partner LU Definition for CICS/VSE

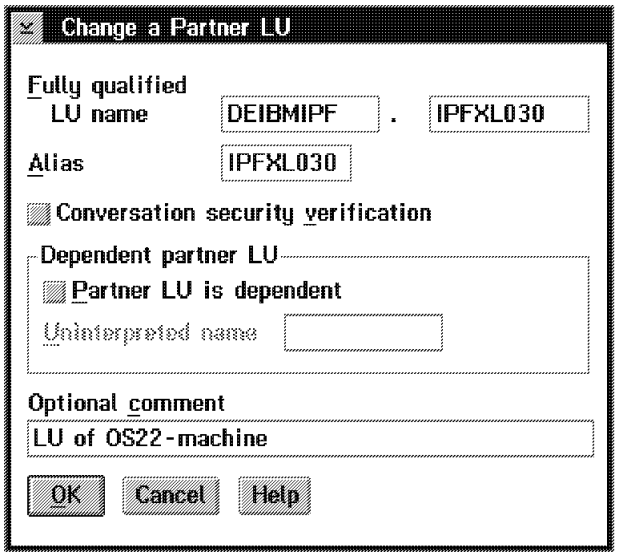

Figure 42. Partner LU Definition for Workstation 2 (OS22)

The key fields are:

- ***Fully qualified LU name.*** The network ID and the partner LU name.

  DEIBMIPF is the ID of the network on which both partner LUs reside.

  CICSSA22 and IPFXL030 are the names of the partner LUs residing on the network whose name is in the previous field.

- ***Alias.*** CICSSA22 and IPFXL030 are the alternative names or alias names by which transaction programs may refer to the LUs.

5.2.4.3 Defining SNA Session Characteristics for OS2S

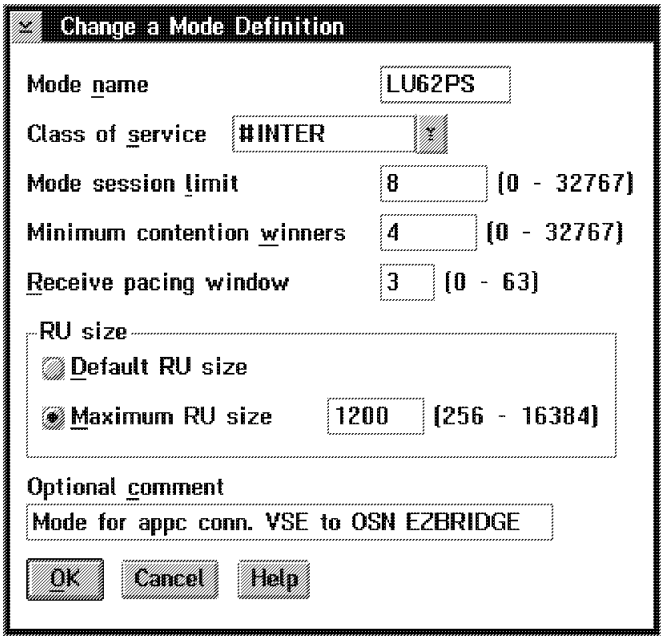

Figure 43. Mode Definition Panel

The key fields are:

- ***Mode Name.*** LU62PS is the name of the mode that contains the properties for OS2S's LU 6.2 sessions. It must match the Mode Name in OS2S's corresponding message channel definitions (for example in Figure 55 on page 72) as well as the DLOGMOD specification in VSE/VTAM's SWNET major node definition (see Figure 11 on page 25).
- ***Class of service.*** #inter is for interactive communications.
- ***Mode Session Limit.*** The maximum number of sessions that can be active at the same time for the LUs using this mode.
- ***Minimum contention winners.*** The minimum number of sessions in which you want a logical unit (LU) in this mode to win in a contention with a partner LU. We have specified four winners for OS2S, and in the host CICS/VSE session definitions in Figure 19 on page 35, we have specified four winners for the host.
- ***Receive Pacing Window.*** The receiving node rate sets the pace for message transmission.
- ***Maximum RU size.*** The largest RU size we expect to be transferred on sessions using this mode.

5.2.4.4 Defining Transaction Programs

Selecting 'Transaction program definitions' from the 'SNA Features List' panel allows you to define transaction programs and to list the programs which already have been defined.

ezBRIDGE on OS/2 provides two CM/2 transaction programs, the MCAs, which are started as soon as CM/2 receives a request from a remote ezBRIDGE system:

- ***MCAS.EXE***, the MCA program for SNA environments.
- ***MCAT.EXE***, the MCA program for TCP/IP environments.

CM/2 can also pass a parameter string to its transaction programs. For ezBRIDGE on OS/2 the name of the active queue manager and the name of the message channel is passed to either MCAS.EXE or MCST.EXE.

For OS2S four transaction programs are required as shown in Figure 44:

- two for the communication with ezBRIDGE on VSE/ESA, 'TPVS' and TPSV' and
- two for the communication with ezBRIDGE on OS/2 on OS22., 'TP2S' and 'TPS2'.

For an overview of the naming conventions for the transaction programs and channel definitions please refer to C.5, "Message Channel and MCA Summary" on page 163.

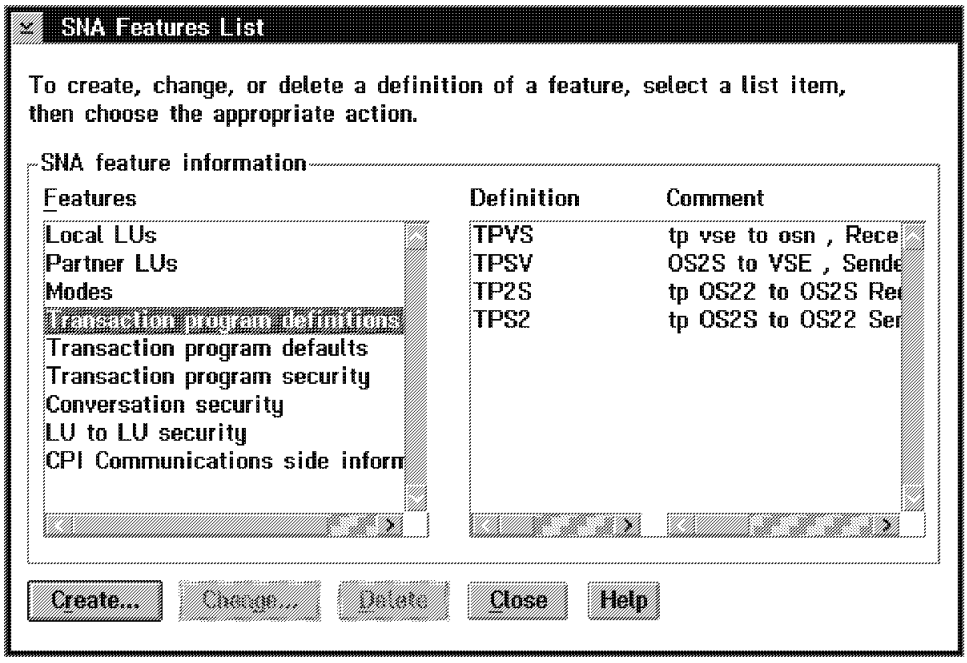

Figure 44. Transaction Program Definition Main Menu

The screens shown in Figure 45 on page 63 to Figure 49 on page 66 show the details of the transaction program definitions required for our MQI environment shown in Figure 3 on page 15.

1. Transaction Program 'TPVS' (Receiver)

Transaction TPVS runs on OS/2 workstation 1 (OS2S) and is the responsible MCA for handling incoming messages from ezBRIDGE on VSE/ESA.

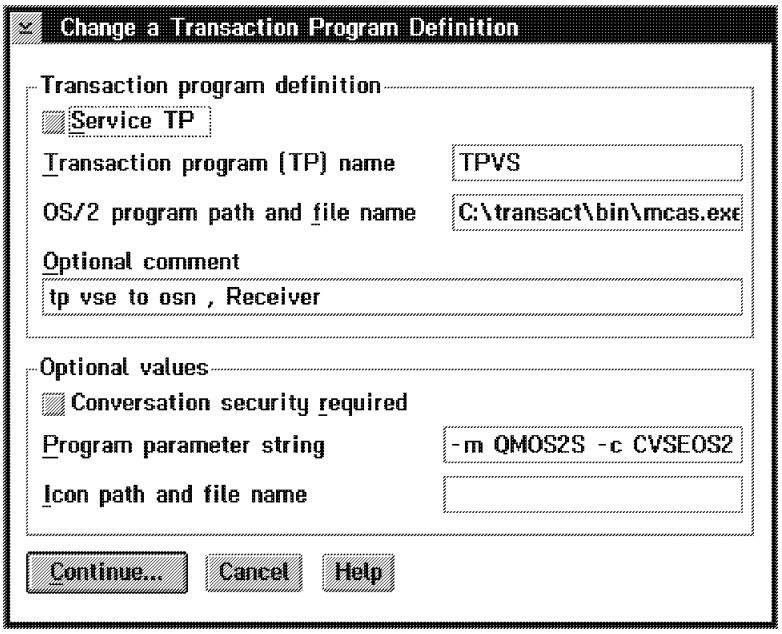

Figure 45. Receiver MCA on OS2S for Messages from ezBRIDGE on VSE/ESA

TPVS defines an MCA for the receiver function. It will be started by the first incoming APPC-ATTACH request and writes MQI messages arriving from other ezBRIDGE systems to the appropriate queue. This receiver MCA should not be manually started.

The key fields are:

- ***Transaction program (TP) name.*** TPVS is the name of the transaction program acting as receiver MCA.
- ***OS/2 program path and the file name.*** C:TRANSACTBIN defines the path to:
- ***MCAS.EXE.*** The OS/2 program started by CM/2 after an incoming APPC-ATTACH request.
- ***Program parameter string.*** '-m QMOS2S' identifies the name of the local, that is OS2S's queue manager, '-c CVSEOS2' identifies the message channel to be used, that is, in this case, the LU 6.2 connection defined for sending messages from ezBRIDGE on VSE/ESA to ezBRIDGE on OS/2. Note that this field is **case sensitive.**

Clicking the '**Continue..** button lets you define additional parameters as shown in Figure 46 on page 64.

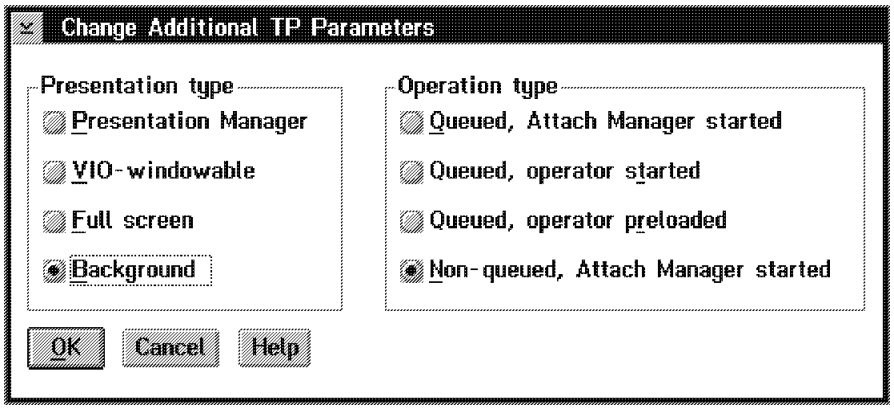

Figure 46. TPVS: Additional TP Parameters

The key fields are:

- ***Background.*** Defines that the transaction program runs in background.
- ***Non-queued, Attach Manager started.*** Defines that multiple instances of the transaction can run at one time. This is the default and recommended option.

2. Transaction Program 'TPSV' (Sender)

Transaction TPSV runs on OS/2 workstation 1 (OS2S) and is the responsible MCA for sending messages to ezBRIDGE on VSE/ESA.

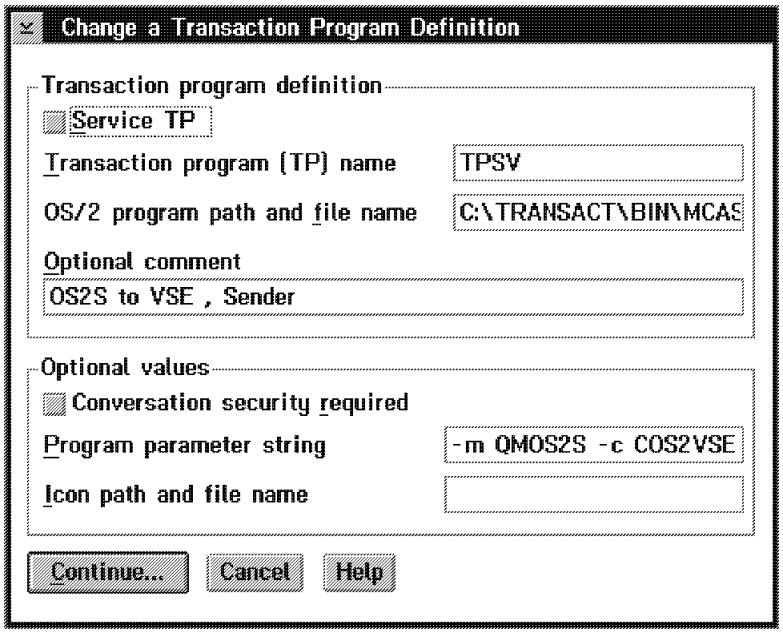

Figure 47. Sender MCA on OS2S for Messages to ezBRIDGE

TPSV defines an MCA for the sender function. Its responsibility is to send messages destined to other ezBRIDGE systems via the corresponding transmission queue using the appropriate message channel. We started TPSV manually by clicking the appropriate icon on the desktop workplace shell.

The key fields are:

- ***Transaction program (TP) name.*** TPSV is the name of the transaction program acting as sender MCA.
- ***OS/2 program path and the file name.*** C:TRANSACTBIN defines the path to:
- ***MCAS.EXE.*** The OS/2 program to be started for ezBRIDGE on OS/2
- ***Program parameter string.*** ′-m QMOS2S′ identifies the name of the local, that is OS2S′s queue manager, ′-c COS2VSE′ identifies the message channel to be used, that is, in this case, the LU 6.2 connection defined for sending messages from ezBRIDGE on OS/2 to ezBRIDGE on VSE/ESA. Note that this field is **case sensitive.**

Clicking the ′**Continue..** button lets you define the same additional parameters as shown in Figure 46 on page 64.

3. Transaction Program ′TP2S′ (Receiver)

Transaction TP2S runs on OS/2 workstation 1 (OS2S) and is the responsible MCA for handling incoming messages from ezBRIDGE on OS/2 on OS/2 workstation 2 (OS22).

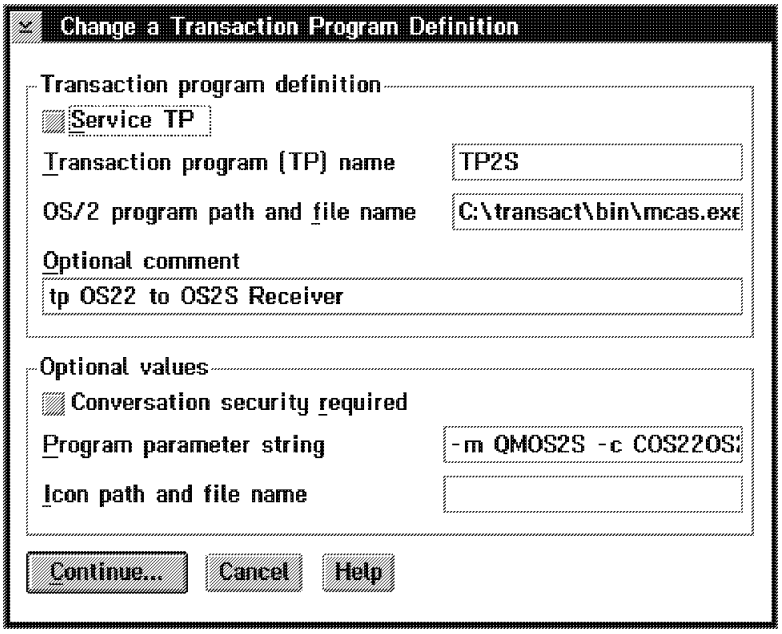

Figure 48. Receiver MCA on OS2S for Messages from OS22

The definitions for MCA TP2S are equivalent to the ones for MCA TPVS with the exception that message channel ′COS22OS2S′ is used to connect to OS22.

4. Transaction Program ′TPS2′ (Sender)

Transaction TPS2 runs on OS/2 workstation 1 (OS2S) and is the responsible MCA for sending messages to ezBRIDGE on OS/2 on OS22.

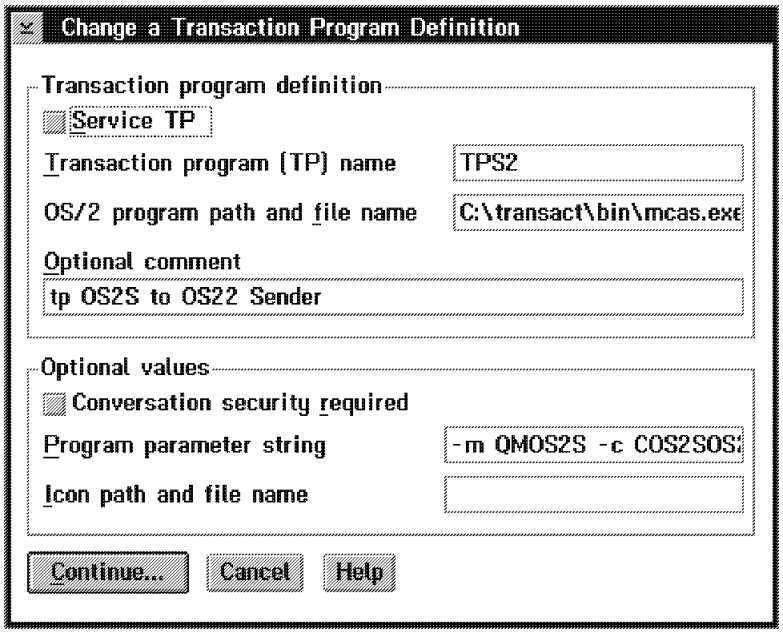

Figure 49. Sender MCA on OS2S for Messages to OS22

The definitions for MCA TPS2 are equivalent to the ones for MCA TPSV with the exception that message channel 'COS2SOS22' is used to connect to OS22.

5.3 ezBRIDGE on OS/2 Customization for Workstation 1 (OS2S)

Important Note

1. All parameters used in ezBRIDGE customization are ***CASE SENSITIVE***.
2. All OS/2 panels showing ezBRIDGE definitions in this chapter **DISPLAY** the parameters which had been entered during the definition step.
3. All definitions for ezBRIDGE are done by using the **MQM** program. We used an icon to start MQM (see 5.4, "ezBRIDGE on OS/2 Operation" on page 75).

This section provides the required definitions regarding ezBRIDGE queues and their management within OS/2. The installation of the product itself is described in detail in *Examples of Using MQSeries on S/390, RISC System/6000, AS/400 and PS/2, GG24-4326*.

As described in 4.4, "ezBRIDGE on VSE/ESA Customization" on page 37, ezBRIDGE can be used locally or in a distributed MQI environment.

In order to exchange messages with ezBRIDGE on VSE/ESA and OS/2 workstation 2 (OS22) as shown in Figure 3 on page 15 and Figure 4 on page 17 we defined:

1. a queue manager named **QMOS2S**
2. a local queue named **FROMVSE**
3. two remote queues named **TOOS22** and **TOVSE**
4. two transmission queues named **QMOS2S** and **QMVSE**

5. two message channels for sending messages named **COS2SOS22** and **COS2SVSE**
6. two message channels for receiving messages named **COS22OS2S** and **CVSEOS2S**

All ezBRIDGE resources mentioned in the above list were created and defined using the configuration and appropriate subsequent options in ezBRIDGE on OS/2's administrator program **MQM**.

5.3.1 Defining the Queue Manager

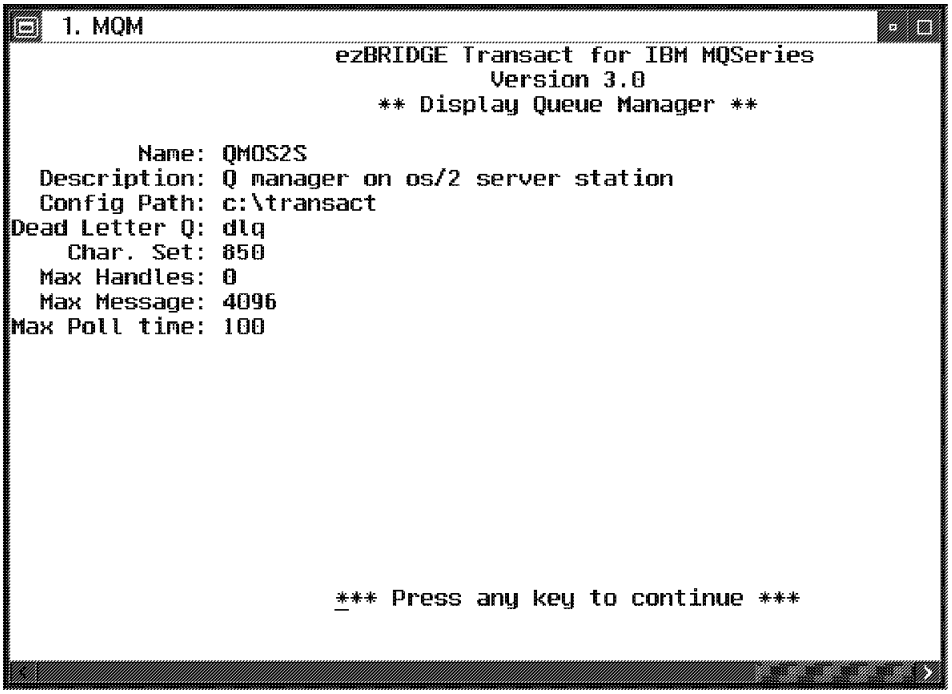

Figure 50. ezBRIDGE Queue Manager Definition

The key fields are:

- ***Name.*** 'MQOS2S' is the name of the queue manager on OS2S. On each OS/2 system only one ezBRIDGE queue manger can be active.
- ***Config Path.*** 'c:\transact' specifies the path for ezBRIDGE's files.
- ***Dead Letter Q.*** 'dlq' is the name of ezBRIDGE on OS/2's dead letter queue.
- ***Char. Set.*** '850' is the multilingual ASCII set.
- ***Max Handles.*** This parameter is currently not used.
- ***Max Message.*** Maximum message length in bytes that QMOS2S can handle.
- ***Max Poll Time.*** Time in milliseconds between polls. We used the default.

5.3.2 Defining a Local Queue

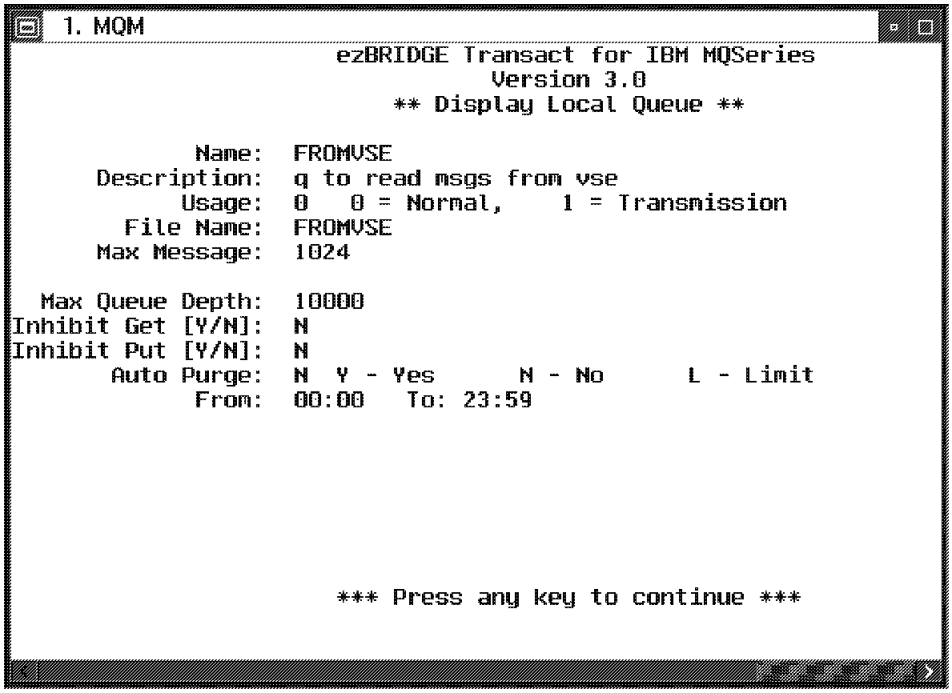

Figure 51. ezBRIDGE on OS/2 Local Queue Definition

The key fields are:

- ***Name.*** ′FROMVSE′ is the name of the local queue on OS2S destined to receive messages from ezBRIDGE on VSE/ESA. It has a corresponding remote queue definition on ezBRIDGE on VSE/ESA, TOOS2 (see Figure 27 on page 45).
- ***Usage.*** For a local queue ″0″ (zero) must be defined.
- ***File Name.*** We used a mnemonic name which is related to the queue′s function and is the same as the queue name.
- ***Max Message.*** 1024 bytes is the maximum message size for this queue. It can be equal to or smaller than the max message size specification in the queue manager definition.
- ***Max Q Depth.*** This is the max number of records this queue can hold.
- ***Inhibit Get/Put.*** Lets you enable or disable reading from or writing to this queue.

5.3.3 Defining Remote Queues

On OS2S we defined two remote queues, TOOS22 and TOVSE as illustrated in Figure 3 on page 15 and Figure 4 on page 17.

5.3.3.1 Remote Queue TOOS22 for Workstation 2 (OS22)

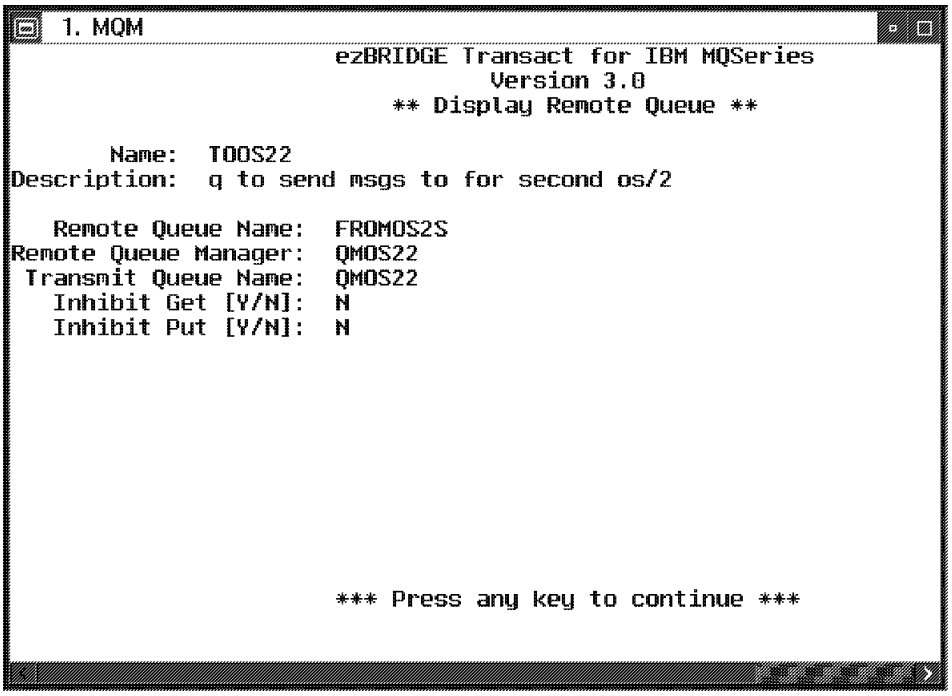

Figure 52. ezBRIDGE Remote Queue Definition

The key fields are:

- ***Name.*** 'FROMOS2S' is the name of the remote queue on OS22, for messages from OS2S. On OS22, FROMOS2S is a local queue.
- ***Remote Queue Manager.*** 'QMOS22' identifies the queue manager on the destination system, OS22.
- ***Transmit Queue Name.*** 'QMOS22' is the name of OS2S's transmission queue for messages which are being sent to OS22.

5.3.3.2 Remote Queue TOVSE for VSE/ESA

A second remote queue, TOVSE which uses transmission queue QMVSE was defined for messages to ezBRIDGE on VSE/ESA.

5.3.4 Defining Transmission Queues

The definition of a transmission queue corresponds exactly to the one of a local queue with the exception of the 'usage' field as shown below.

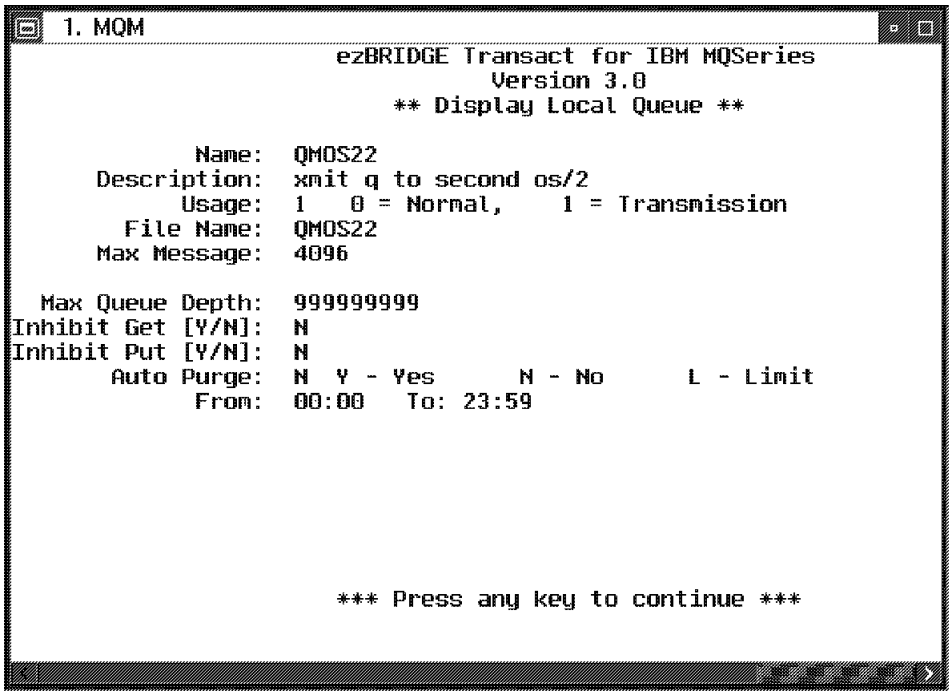

Figure 53. ezBRIDGE Transmission Queue Definition

The key fields are:

- ***Usage.*** '1' defines a transmission queue for outbound messages.
- ***File Name.*** 'QMOS22' identifies the file where the transmission queue records are physically stored. ezBRIDGE on OS/2 adds a file extension of ".que" to the file name.
- ***Max Message.*** This transmission queue can handle messages with a maximum length of 4096 bytes.
- ***Max Queue Depth.*** Using the default of 999999999 means that there is no limit with regard to the number of messages that can be stored in QMOS22.

In the same way we defined a second transmission queue, QMVSE for messages to ezBRIDGE on VSE/ESA.

5.3.5 Defining Message Channels

Creating channels involves two screens. The first is the same for all channels and allows for entering generic parameters, the second varies according to the transport protocol selected for the channel (SNA in our case).

5.3.5.1 Sender Channels COS2VSE and COS2SOS22

1. **COS2VSE**

   The two screens below show the definition of the sender channel COS2VSE which, together with its counterpart COS2VSE on ezBRIDGE on VSE/ESA, provides a transfer link for messages between OS2S and the VSE/ESA host.

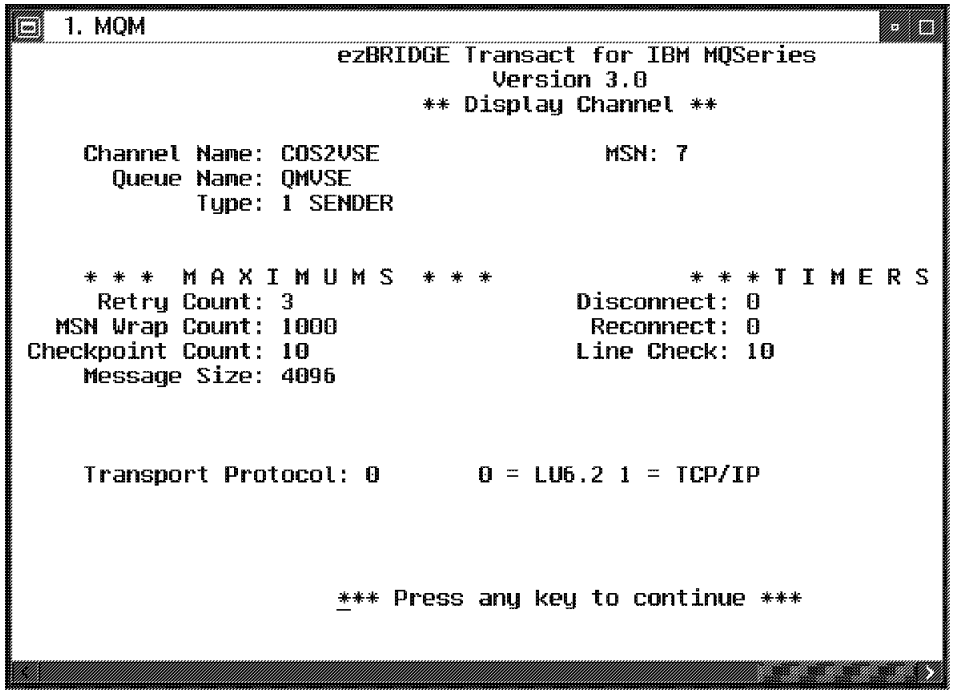

Figure 54. Sender Message Channel for OS2S to VSE/ESA (1 of 2)

The key fields are:

- ***MSN.*** Value for the Message Sequence Number. Set to ′1′ at creation time, this value is maintained by ezBRIDGE on OS/2 and synchronized with the corresponding remote message channel connected to this channel.
- ***Queue Name.*** The name of the associated transmission queue. This field is not used by a receiver channel
- ***Type.*** ′1′ determines a sender channel.
- ***Message Size.*** We defined a maximum message size of 4096 bytes. During our tests we found that it is possible to exchange messages between channels with different values for ′Message Size′. The two communicating MCA′s ′negotiate′ a message length acceptable by both partners, that is the smaller message size.
- ***Transport Protocol.*** ′0′ (zero) indicates SNA LU 6.2. The only other choice is ′1′ for TCP/IP.

The second screen refers to transport protocol parameters:

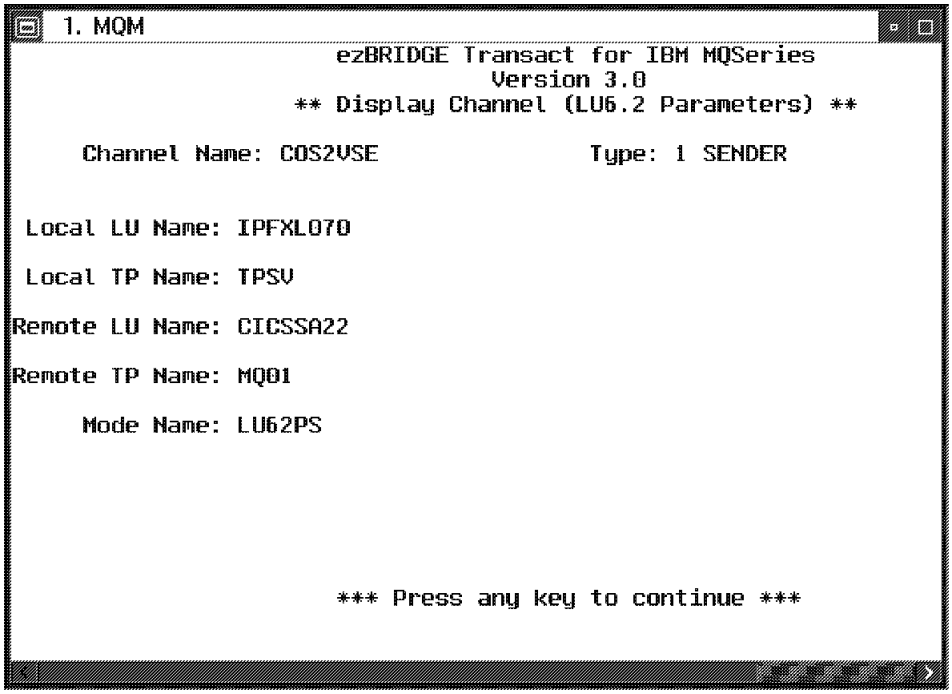

Figure 55. Sender Message Channel for OS2S to VSE/ESA (2 of 2)

- ***Local LU Name.*** 'IPFXL070' is the LU name for OS/2 workstation 1 (OS2S).
- ***Local TP Name.*** 'TPSV' is the name of the transaction program representing the MCA for this channel. The name must match the corresponding transaction program definition in the SNA features during CM/2 customization (see Figure 47 on page 64).
- ***Remote LU Name.*** 'CICSSA22' is the LU name of the partner LU, CICS/VSE in our case. It must match the corresponding partner LU definition during CM/2 customization (refer to Figure 41 on page 60).
- ***Remote TP Name.*** 'MQ01' identifies the MCA for receiving messages on ezBRIDGE on VSE/ESA
- ***Mode Name.*** 'LU62PS' is the name of the SNA mode used for sessions over this channel. It must match the mode definition in the corresponding CM/2 customization (refer to Figure 43 on page 61).

2. **COS2SOS22**

   Channel COS2SOS22 is the second sender channel which, together with its homonymous counterpart COS2SOS22 on OS/2 workstation 2 (OS22) provides a transfer link for messages between the two OS/2 platforms.

   The definitions correspond to the ones shown in the screens for sender channel COS2VSE above, we just specified the appropriate names to reflect OS22 as receiver:

   - 'QMOS22' for the transmission queue for OS22
   - 'TPS2' as local transaction program name for TPOS2S's MCA to send messages to OS22
   - 'IPFXL030' as remote LU name of the partner LU on OS22

- 'TPS2' as remote transaction program name for OS22's MCA to receive messages from OS2S (see Figure 69 on page 86)

5.3.5.2 Receiver Channels CVSEOS2 and COS22OS2S

1. **CVSEOS2**

   The two screens below show the definition of the receiver channel CVSEOS2 which, together with its counterpart CVSEOS2 on ezBRIDGE on VSE/ESA, provides a transfer link for messages between VSE/ESA and workstation 1 (OS2S).

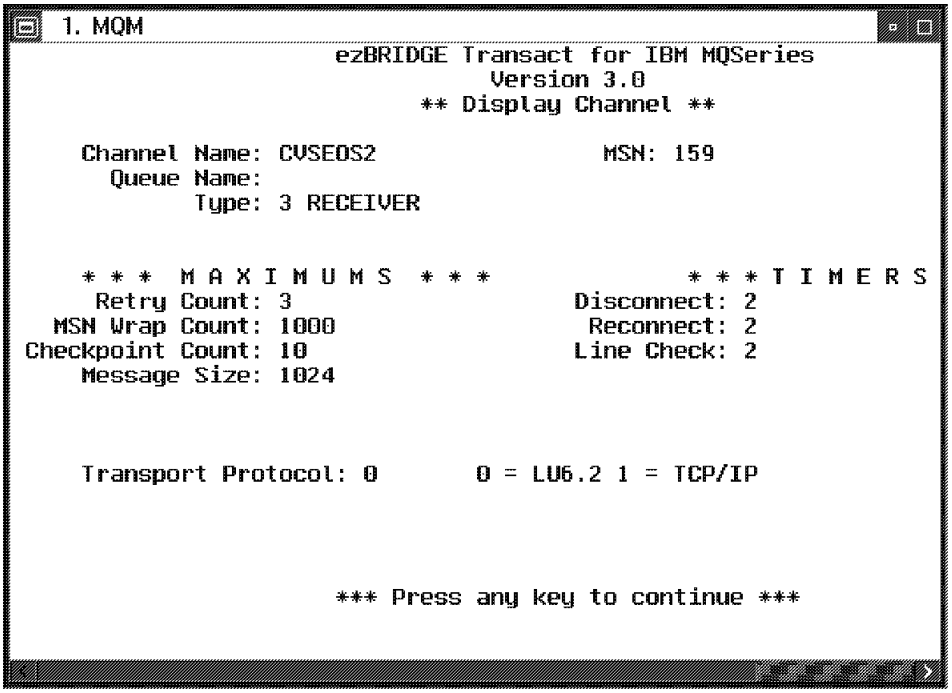

Figure 56. Receiver Message Channel for OS2S from VSE/ESA (1 of 2)

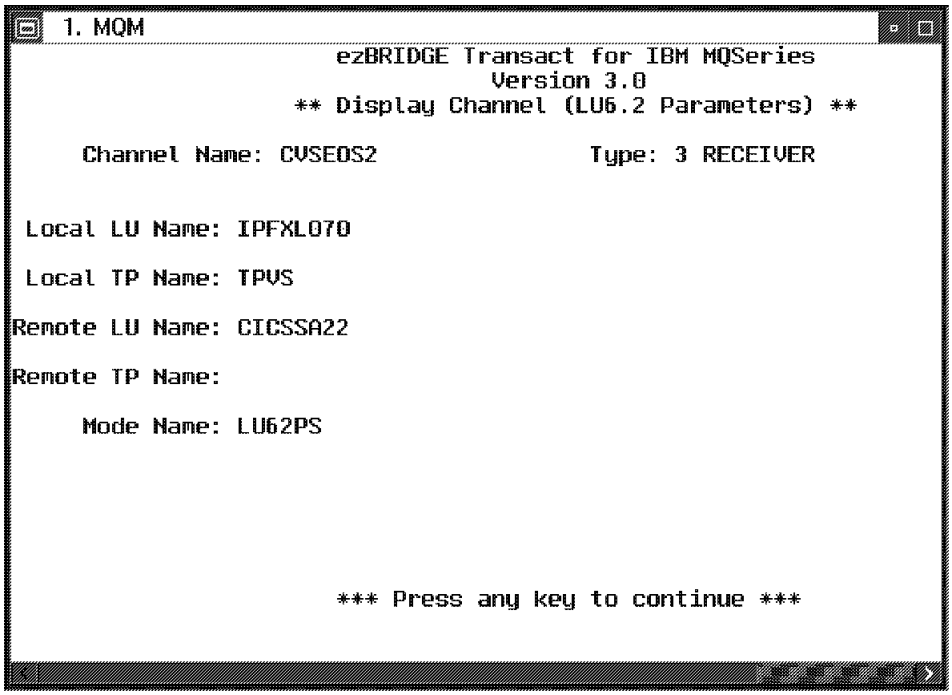

Figure 57. Receiver Message Channel for VSE/ESA to OS2S (2 of 2)

The values we entered into the different fields correspond to the ones explained in 5.3.5.1, “Sender Channels COS2VSE and COS2SOS22” on page 71, we just specified the appropriate values and names to reflect OS2S as receiver:

- type ′3′ to define a receiver channel
- ′TPVS′ as local transaction program name for TPOS2S′s MCA to receive messages from VSE/ESA
- ′CICSSA22′ as remote LU name of the partner LU on VSE/ESA
- ′MQ03′ as remote transaction program name for VSE/ESA′s MCA to send messages to OS2S

During our tests we found out that the field ′Remote TP name′ for a receiver channel definition on a VSE/ESA system must be blank, whereas the value on a receiver channel on ezBRIDGE on OS/2 is ignored.

2. **COS22OS2S**

   Channel COS22OS2S is the second receiver channel which, together with its homonymous counterpart COS22OS2S on OS/2 workstation 2 (OS22) provides a transfer link for messages between the two OS/2 platforms.

   The definitions correspond to the ones shown in the screens for receiver channel CVSEOS2 above, we just specified the appropriate names to reflect OS2S as receiver:

   - ′TP2S′ as local transaction program name for TPOS2S′s MCA to receive messages from OS22
   - ′IPFXL030′ as remote LU name of the partner LU on OS22

5.4 ezBRIDGE on OS/2 Operation

We recommend to define and use ICONs on the OS/2 Desktop to invoke the following MQI functions:

1. **MQ-Demon**

   The MQ-Demon is a server program which is required by the MCA. This process must be running before any MCA is started. Using an ICON to start the demon path and filename in the icon setup screen should define *C:BINMCAMD.EXE*. Alternatively the program can be started by entering the same command in an OS/2 window or fullscreen.

2. **MQ-Main**

   MQ-Main represents the ezBRIDGE on OS/2 administrator program MQM. The program can be started via an icon by having defined *C:BINMQM.EXE* as path and filename in an OS/2 window or fullscreen or by entering this command manually on an OS/2 command line. The MQM program provides a menu-driven administration facility to:

   - Create, modify and display queue managers, queues, and message channels
   - Activate/deactivate message channels and MCAs
   - Browse queue records
   - Start/stop monitoring functions
   - Start/stop tracing
   - Synchronize message sequence numbers between different ezBRIDGE systems.

3. **Send-to-VSE**

   This starts the MCA responsible for sending messages to ezBRIDGE on VSE/ESA. It can be started with *C:TRANSACTMCAS.EXE* as path and filename in the icon setup including ′-m QMOS2S -c COS2VSE′ as parameters to identify the appropriate destination queue manager and message channel.

   To start the sender MCA manually the command is as follows: C:TRANSACTMCAS -m QMOS2S -c COS2VSE.

4. **Send-to-OS22**

   This starts the MCA responsible for sending messages to OS22. Manual start is done via the command ′*C:TRANSACTMCAS* including ′-m QMOS2S -c COS2SOS22′ as parameters to identify the appropriate destination queue manager and message channel. To start the program by clicking an icon the path and filename field is *C:TRANSACTMCAS.EXE* and the parameter field must include ′-m QMOS22 -c COS2SOS22′.

5.5 ezBRIDGE on OS/2 Customization for Workstation 2 (OS22)

This section describes the steps required to prepare the ezBRIDGE environment for the secondary PS/2 called **OS22** in our sample configuration as depicted in Figure 4 on page 17.

In our network, this secondary PS/2 will communicate directly with another PS/2 and only **indirectly** with the host VSE/ESA system. All the message traffic to and

from this system will be routed through the primary PS/2 named **OS2S**. Our OS/2 Communication Manager will be customized for this connection only. It will **not** have any reference to the VSE/ESA host system.

The following aspects of system OS22′s customization are discussed:

- Communication Manager customization
- ezBRIDGE Customization using the MQM program

5.5.1 Communication Manager Customization

Before CM/2 can use the token-ring, the prerequisite LAPS software had to be installed. This was done as described in chapter 5.1, “LAPS Installation and Customization” on page 51. The only difference is that the system OS22 machine′s network adapter address is 400010101003.

For our APPC (LU6.2) link, the following definitions are required in the CM configuration files:

1. One **DLC Adapter Parameters** definition, which specifies Data Link Control characteristics of the token-ring adapter.
2. One **Local node characteristics** definition, which specifies characteristics that are common to all APPC users on the workstation.
3. One **Connection** definition.
4. A **Local Logical Unit** definition to describe this workstation as a Logical Unit available to the network.
5. A **Partner Logical Unit** definition for the PS/2 with which we are going to communicate.
6. A **MODE** definition to specify sets of session properties that are used in binding APPC sessions.
7. A **Transaction program** definition for every local transaction that can be invoked in an outbound request to a remote system or an inbound request from a remote system.

The following are the relevant parameters in the Communications Manager Profile List Sheet in the C/M configuration definition. They describe parameters used in our scenario to define the token-ring APPC connection to the primary ezBRIDGE OS/2 system named **OS2S** in our scenario.

5.5.1.1 Configure DLC Token-Ring

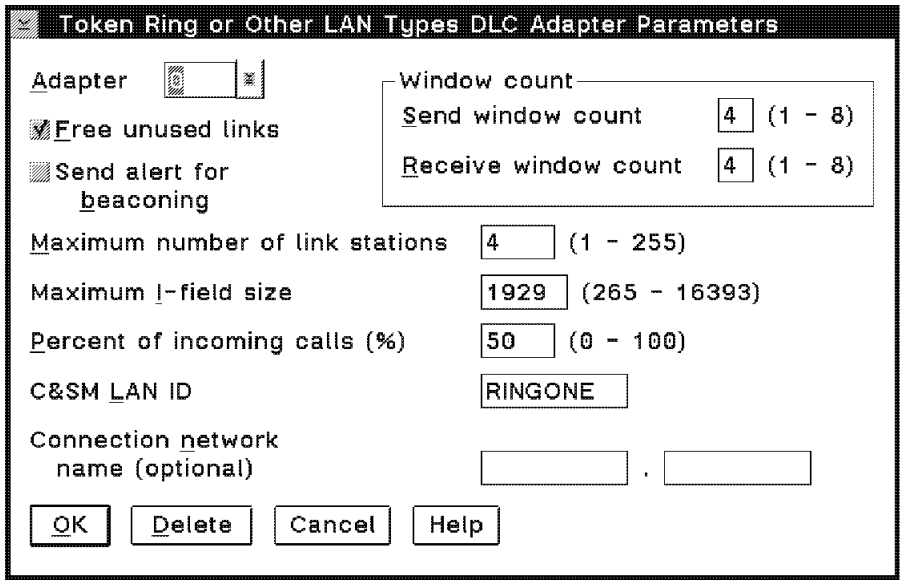

Figure 58. Token-Ring DLC Adapter Parameters

You can take most defaults here. You should enter a name for the following parameter:

- ***C&SM LAN ID.*** To identify the ring - used by system management products.

5.5.1.2 Defining Local Node Characteristics

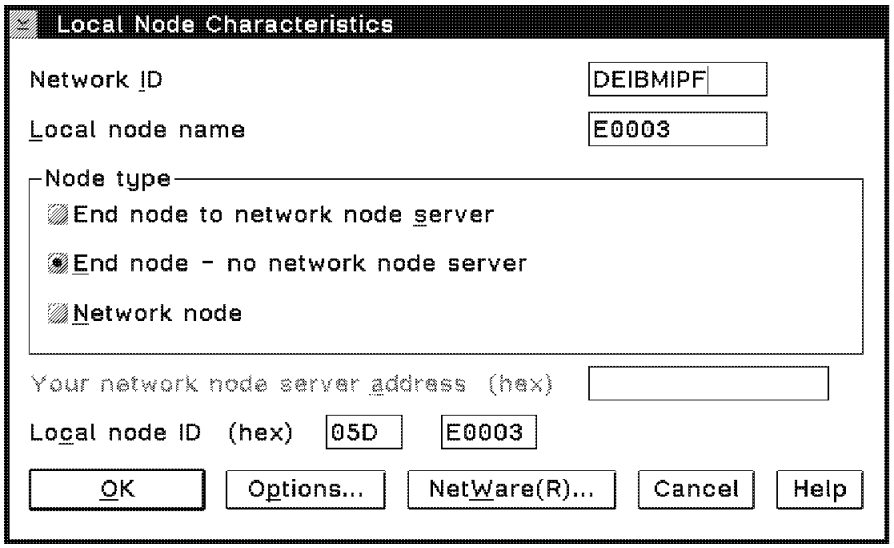

Figure 59. Local Node Characteristics

The key fields are:

- ***Network ID.*** DEIBMIPF is the name of the network to which this PS/2 belongs.
- ***Node type.*** Our PS/2 is an end node accessed by another PS/2 on the same token-ring.
- ***Local Node ID.*** The values 05D E0003 are specified here for potential future connection to a host 37x5. They are not required for our PS/2 to PS/2 connection, but it is recommended they be specified.

Local Node Options On the following window opened with the ′Options′ pulldown, click on the option:
Activate Attach Manager at start up.

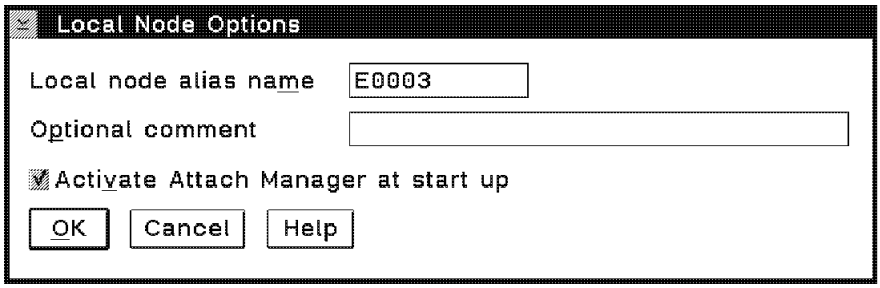

Figure 60. Local Node Options

The key field is:

- ***Activate Attach Manager.*** This is a feature of APPC which manages incoming requests to begin a communication session. In our case, the Attach Manager is started automatically when the Communications Manager is started.

5.5.1.3 Define Connection to a Peer Node

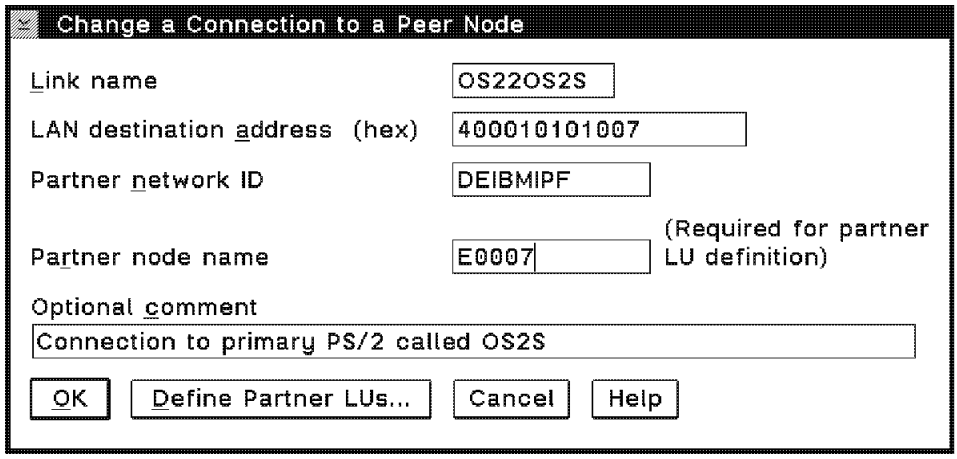

Figure 61. Connection Definition Panel

The key fields are:

- ***Link Name.*** OS22OS2S is the link name we chose for the connection to the PS/2 system which we call OS2S in the ezBRIDGE network.
- ***LAN Destination address.*** 400010101007 is the LAN MACADDRess of the target PS/2 to which we will connect.
- ***Partner network ID.*** DEIBMIPF is the name of the network to which the target PS/2 belongs. It happens to be installed in the same network as our secondary PS/2.
- ***Partner node name.*** E0007 is the local node name specified in the target PS/2's local node characteristics as in Figure 35 on page 55.

5.5.1.4 Defining Optional SNA Features

Figure 62 on page 80 lists the features which may be included in the local configuration of CM/2.

On our secondary ezBRIDGE system, we defined options for the first four features:

- Local LUs
- Partner LUs
- Modes
- Transaction program definitions

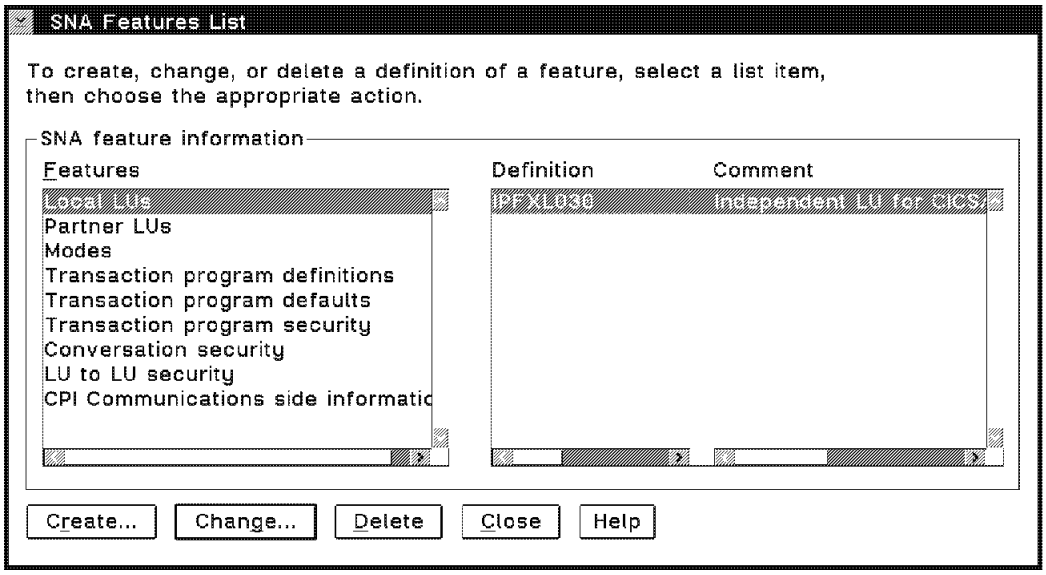

Figure 62. List of Optional SNA Features

Local LU Definition

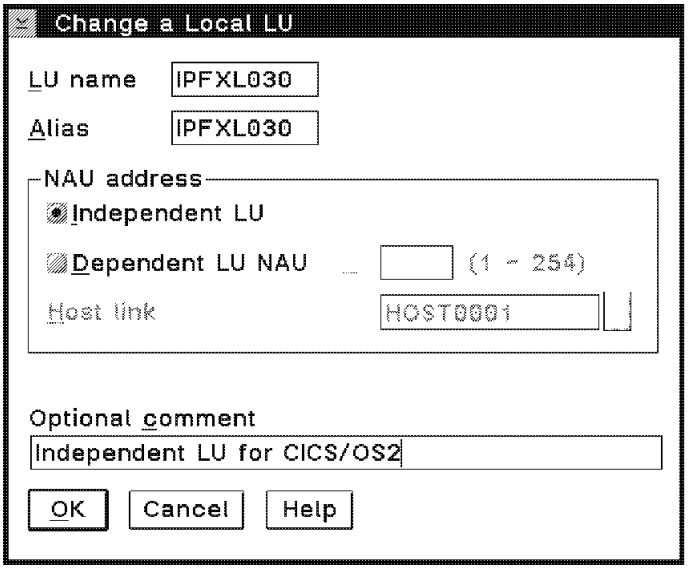

Figure 63. Local LU Definition Panel

The key fields are:

- ***LU Name.*** IPFXL030 is the name of the Logical Unit known by other ezBRIDGE systems as the one which runs this secondary OS/2 system's Queue Manager. It is defined in the ezBRIDGE channel definitions for our OS/2 systems as in Figure 77 on page 91 and Figure 79 on page 92.
- ***Alias.*** IPFXL030 is the alias name by which transaction programs may refer to this LU.
- ***IPFXL030.*** Defined as an independent LU. For more information about independent LUs please refer to Chapter 4, "ezBRIDGE on VSE/ESA Implementation" on page 19.

5.5.1.5 Defining the Partner LU

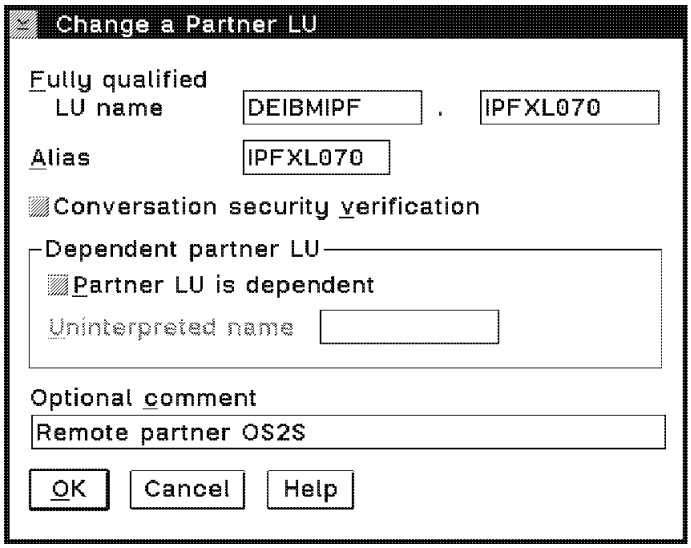

Figure 64. Partner LU Definition to PS/2 System OS2S.

The key fields are:

- ***Fully qualified LU name.*** The network ID and the partner LU name.

  DEIBMIPF is the ID of the network on which our partner LU resides.

  IPFXL070 is the name of our partner LU in the DEIBMIPF network.

- ***Alias.*** IPFXL070 is the alternative name, or alias, by which transaction programs may refer to the partner LU.

5.5.1.6 Defining the Logon Mode

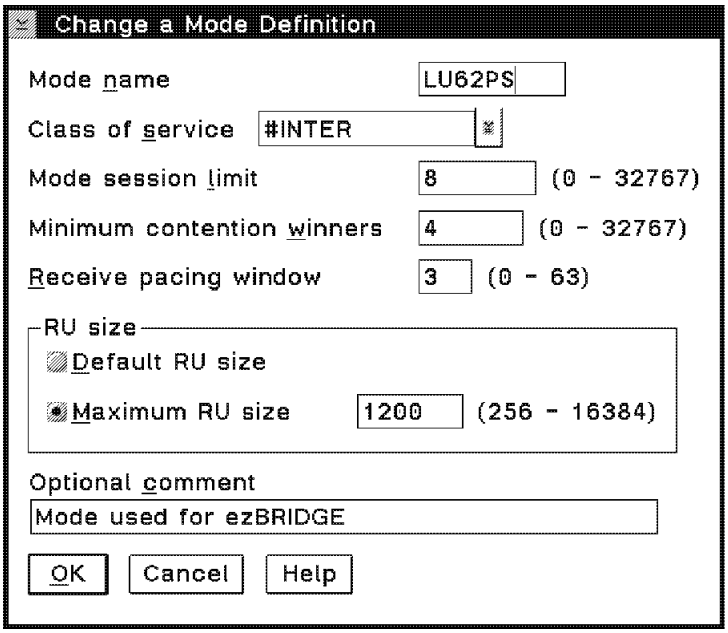

Figure 65. Mode Definition Panel

The key fields are:

- ***Mode Name.*** LU62PS is the name of the mode that contains the session properties for the LU 6.2 session. It must match the Mode Name in the ezBRIDGE channel definition as in Figure 77 on page 91 and Figure 79 on page 92.
- ***Class of service.*** '#inter' is for interactive communications.
- ***Mode Session Limit.*** The maximum number of sessions that can be active at the same time for the LUs using this mode.
- ***Minimum contention winners.*** The minimum number of sessions in which you want a logical unit (LU) in this mode to win in a contention with a partner LU. We have specified four winners for our PS2 end, and four in the primary system's mode definition.
- ***Receive Pacing Window.*** The receiving node rate sets the pace for message transmission.
- ***Maximum RU size.*** The largest RU size we expect to be transferred on sessions using this mode.

5.5.1.7 Defining Transaction Programs

Selecting Transaction program definitions from the SNA Features List allows you to define transaction programs and to see what has already been defined. When CM/2 receives a request from a remote system to start a transaction, it must be told via these menus which program to start locally. ezBRIDGE supplies two programs for this purpose:

- ***MCAS.EXE***, which is the program for a message channel agent (MCA) to be used in an SNA environment.
- ***MCAT.EXE***, which is the program for a message channel agent (MCA) to be used in a TCP/IP environment.

CM/2 can also pass a parameter string to this called program. The name of the Queue Manager and the name of the ezBRIDGE channel definition has to be passed to the TP program (MCA).

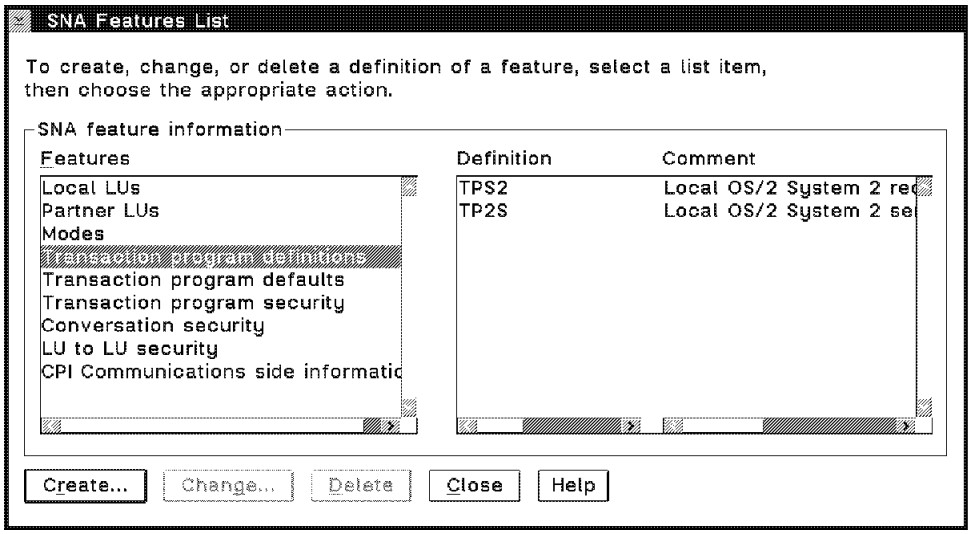

Figure 66. Transaction Program Definition Main Menu

1. Transaction Program 'TPS2' (Receiver)

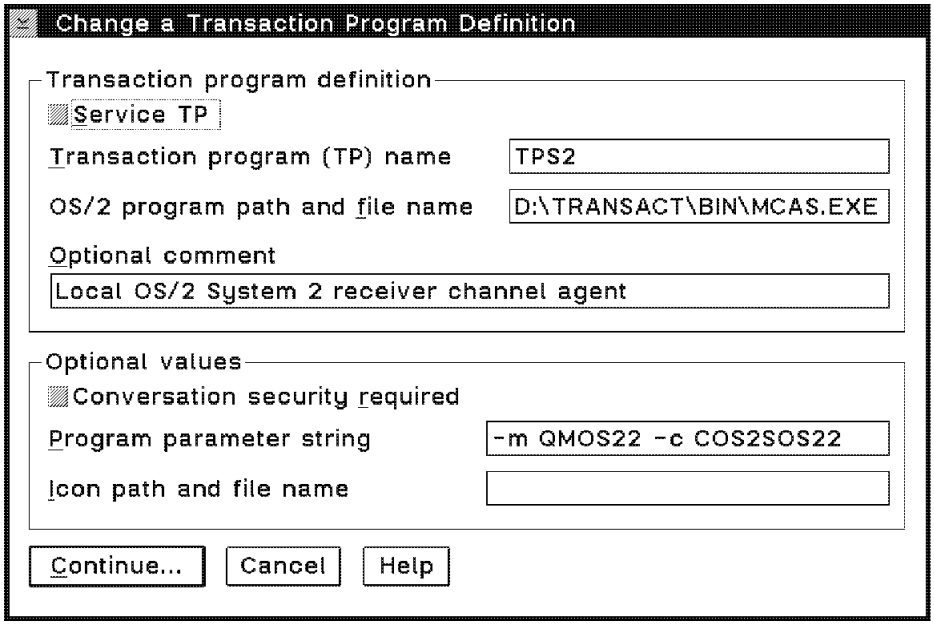

Figure 67. Receiver Program Definition

The TPS2 TP definition starts an MCA for the receiver function. It will be started on the first incoming APPC attach request and will handle the storage of the incoming message queue message on the appropriate queue. Do not start the receiver MCA manually.

The key fields are:

- ***Transaction program (TP) Definition.*** TPS2 is the name of a transaction which CM/2 may receive.

- ***OS/2 program path and the file name.*** D:TRANSACTBIN defines the path to:
- ***MCAS.EXE.*** The OS/2 program which CM/2 starts to initiate the ezBRIDGE channel agent functions.
- ***Program parameter string.*** This field is **case sensitive.** The parameters passed to the program MCAS, specify which queue manager and which MCA to start. In our case, **QMOS22** is the name of our secondary system's queue manager as specified by the MQM function in Figure 70 on page 87. COS2SOS22 is the name of our receiver message channel as specified in Figure 78 on page 92.

Additional TP Parameters

Clicking on 'Continue' from Figure 67 on page 84 results in the following panel:

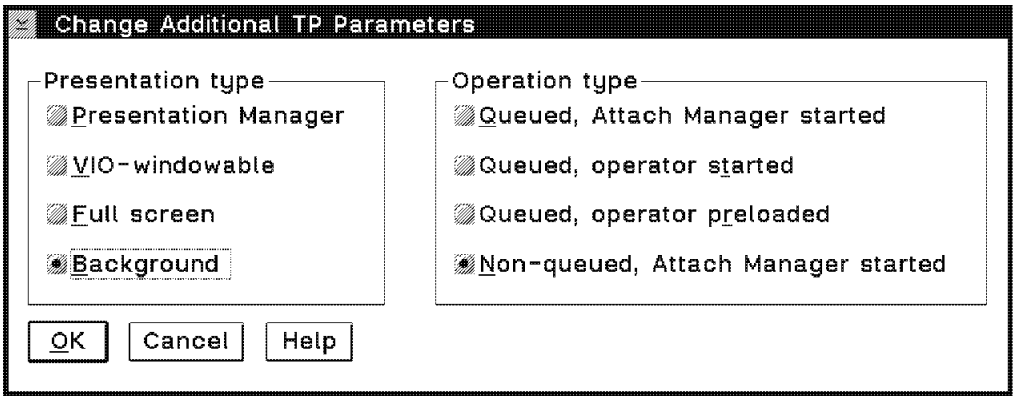

Figure 68. Transaction Program Definition (Additional TP Parameters)

The key fields are:

- ***Background.*** Defines that the transaction program runs in background.
- ***Non-queued, Attach Manager started.*** This is the default and recommended option.

2. Transaction Program 'TP2S' (Sender)

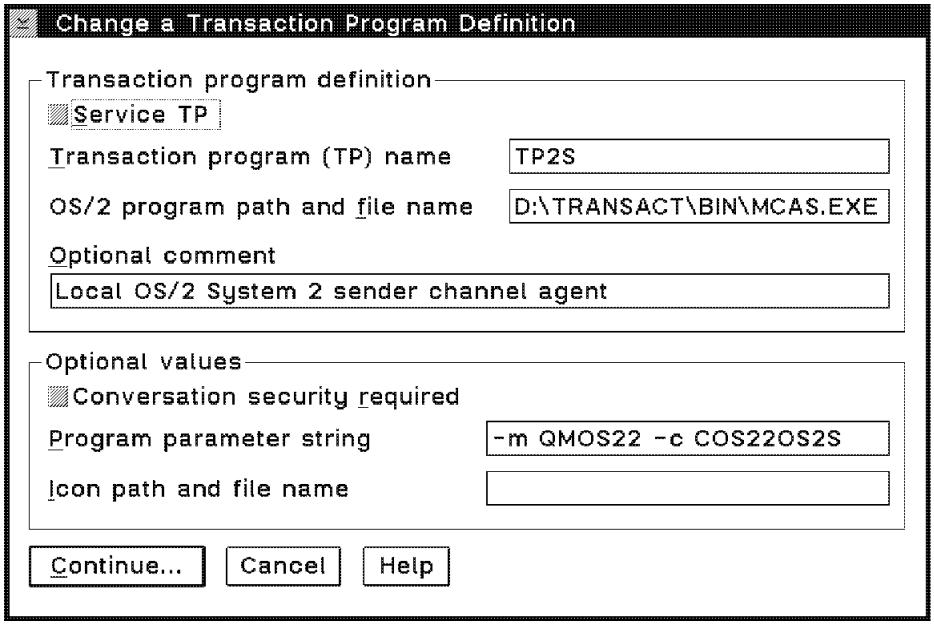

Figure 69. TP2S Transaction Program Definition

The transaction associated with the MCA sender is defined to the Coms Manager in a fashion similar to the receiver. The MCAS program is specified, the parameters passed are the local queue manager name and the sender channel name.

Additional TP Parameters for TP2S

- The parameters are the same as those shown in Figure 68 on page 85.

5.5.2 ezBRIDGE Customization with MQM Program

ezBRIDGE must be customized to reflect the network of other message queue systems and their queues. The following section will describe the ezBRIDGE parameters used to define the queues and channels on the secondary OS/2 system called **OS22** in Figure 4 on page 17.

As depicted there, the OS22 machine was defined to have one message channel for sending messages named *'COS22OS2S'* and one message channel for receiving messages named *'COS2SOS22'* communicating with the system called **OS2S**.

In the diagram, you will also notice that there are five queues defined:

- two local queues, FROMOS2S and FROMVSE
- two remote queues, TOOS2S and TOVSE
- one transmission queue, QMOS2S

There is in fact a sixth queue defined, but not shown, called the Dead Letter Queue. The Dead Letter Queue will be used to store messages which our receiver channel accepts, but for which we have no local queue defined. Definition of a Dead Letter Queue is highly recommended on all MQSeries systems.

Figure 70 on page 87 is the queue manager definition for system OS22.

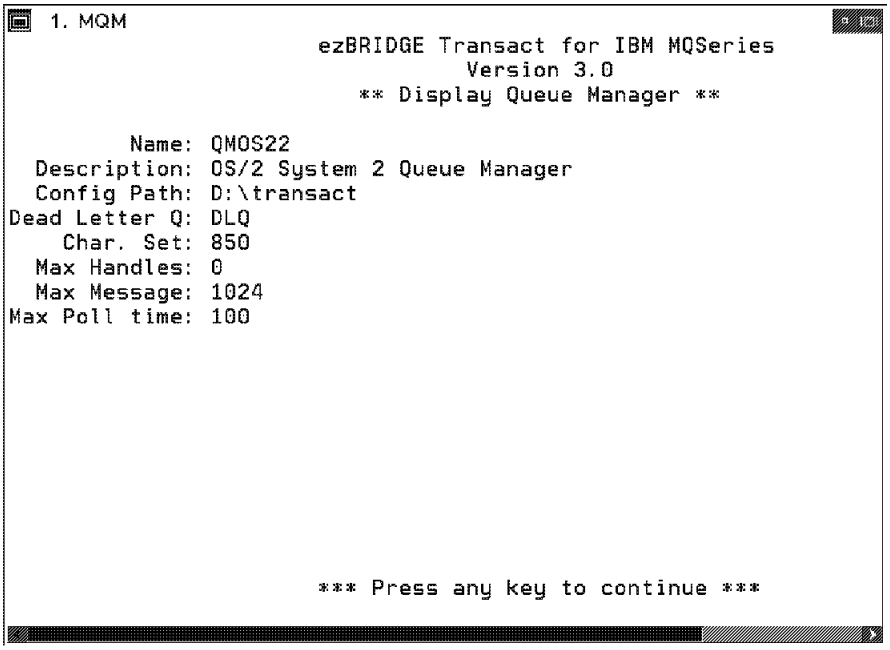

Figure 70. Define Queue Manager for OS/2 System 2 (OS22)

Note the following:

- ***QMOS22*** is the name of this system's queue manager. This name will be used in other ezBRIDGE systems for queues which reside here. It is similar in concept to an RSCS node name.
- ***Max Message*** is specified as 1024 bytes.

Figure 71 on page 88 is the queue definition for system OS22's local queue called **FROMOS2S**. This is the queue to which we expect an application on system OS2S to send messages. On the OS2S system, applications can write to the queue called TOOS22, and the message will be delivered here.

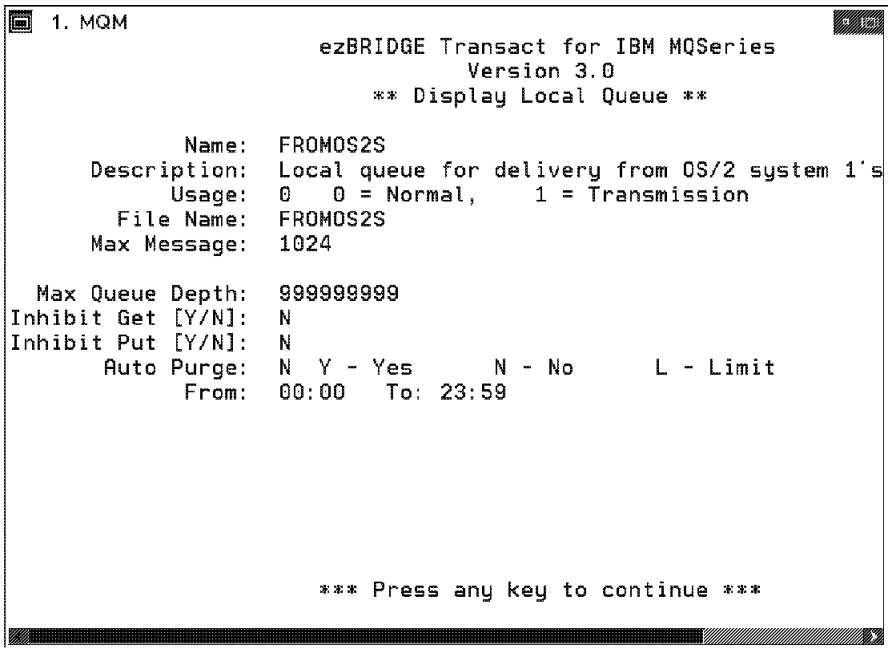

Figure 71. Define Local Queue FROMOS2S on OS/2 System 2 (OS22)

Figure 72 is the queue definition for system OS22's local queue called **FROMVSE**. This is the queue to which we expect an application on the VSE/ESA host to send messages. On the VSE/ESA host, applications can write to the queue called TOOS22, and the message will be delivered here.

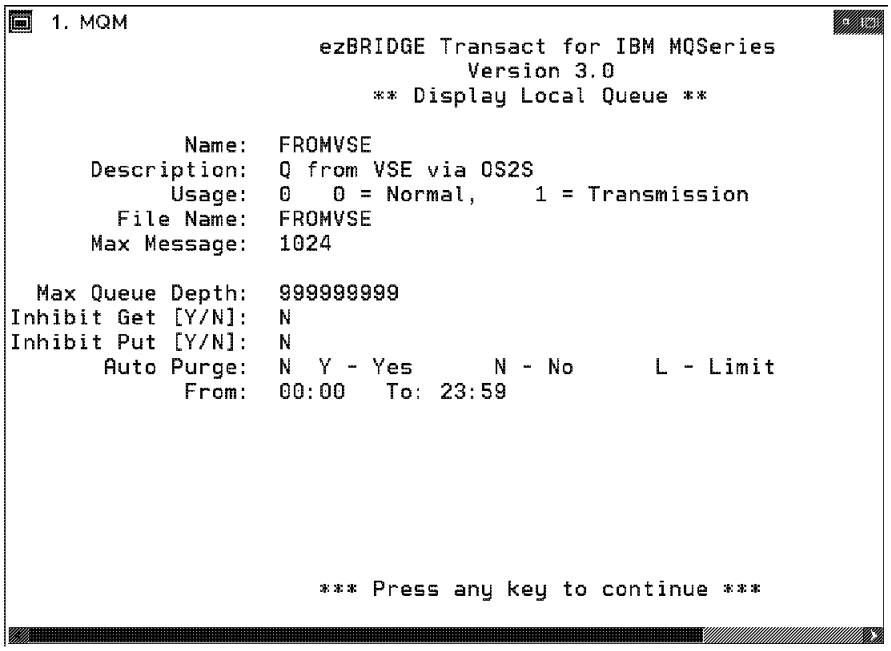

Figure 72. Define Local Queue FROMVSE on OS/2 System 2 (OS22)

Figure 73 on page 89 is the queue definition for system OS22's local transmission queue called **QMOS2S**. This is the queue which will be used to store messages destined for other systems in our network. It is from this queue that our sender channel agent will read and forward messages to the MQSeries

system called QMOS2S. Note that the **destination MQSeries system name (that is the MQM′s name) and this transmission queue name are identical**.

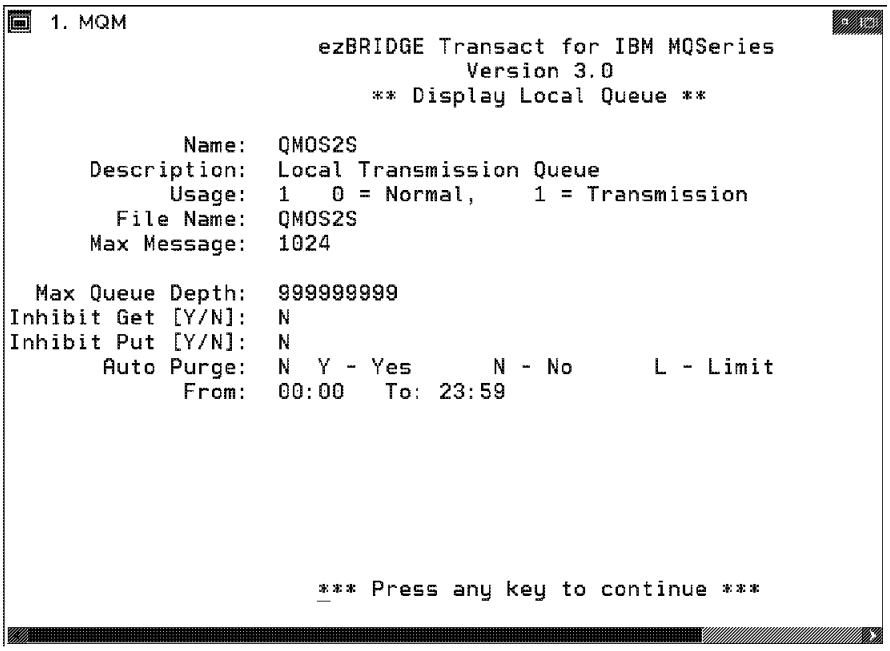

Figure 73. Define Transmit Queue to Primary System (OS2S)

Figure 74 on page 90 is the queue definition for system OS22′s remote queue called **TOOS2S**. When local applications write messages to this queue, ezBRIDGE will store them physically on the transmit queue called QMOS2S. Then the sender MCA will send them to the queue manager associated with this transmit queue. In Figure 76 on page 91 you will see the name of the transmit queue which is associated with our sender channel. When the message is sent over the channel, the receiving system, OS2S, inspects the message header information. It determines that this message is intended for its local queue named FROMOS22.

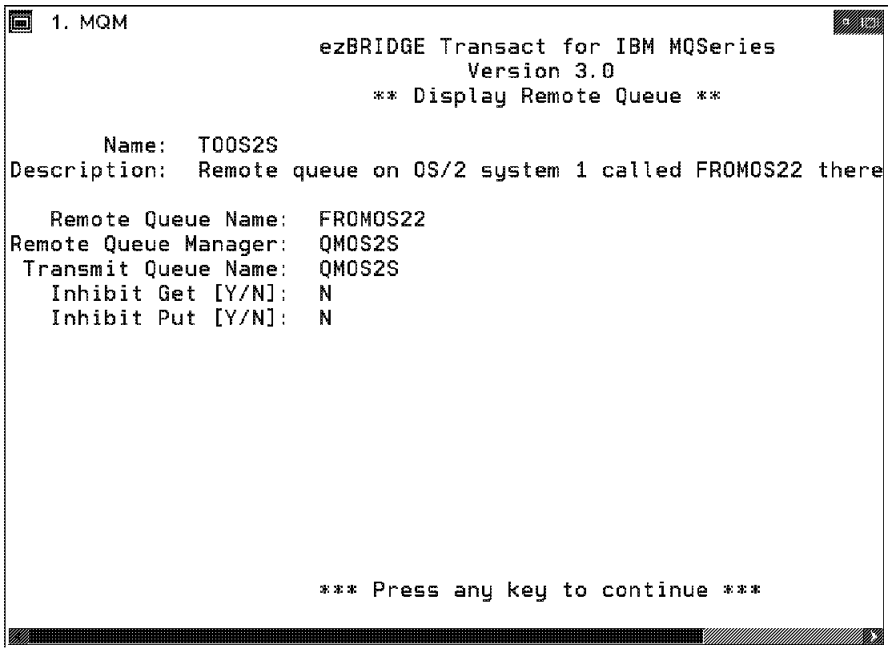

Figure 74. Define Remote Queue TOOS2S

Figure 75 is the queue definition for system OS22′s remote queue called **TOVSE**. When local applications write messages to this queue, ezBRIDGE will store them physically on the transmit queue called QMOS2S. Then the sender MCA will send them to the queue manager associated with this transmit queue. In Figure 76 on page 91 you will see the name of the transmit queue which is associated with our sender channel. When the message is sent over the channel, the receiving system, OS2S, inspects the message header information. It determines that this message is intended for a queue manager named QMVSE. In the QMOS2S system configuration, there is a transmit queue with this name. The QMOS2S manager puts the message on this transmit queue. From there it is sent by its sender MCA to the VSE/ESA host system.

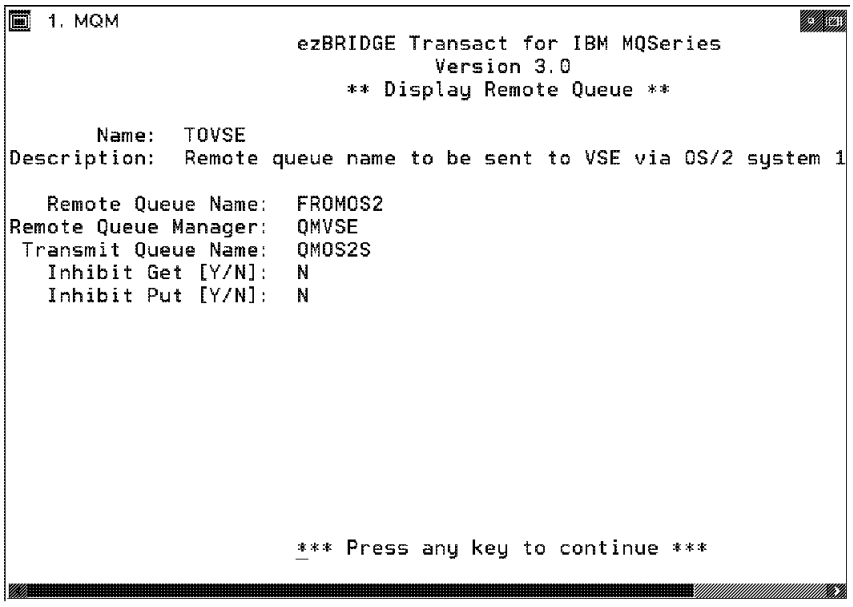

Figure 75. Define Remote Queue TOVSE

Figure 76 on page 91 and Figure 77 on page 91 show the channel definitions for system OS22's sender message channel called **COS22OS2S**. After this channel is started, messages which are placed on the transmission queue **QMOS2S** will be sent over the channel to our primary OS/2 system. In Figure 77 on page 91, you can see the names defined earlier to OS/2 CM/2 for LU 6.2 session establishment. These are described in chapter 5.5.1.4, "Defining Optional SNA Features" on page 79.

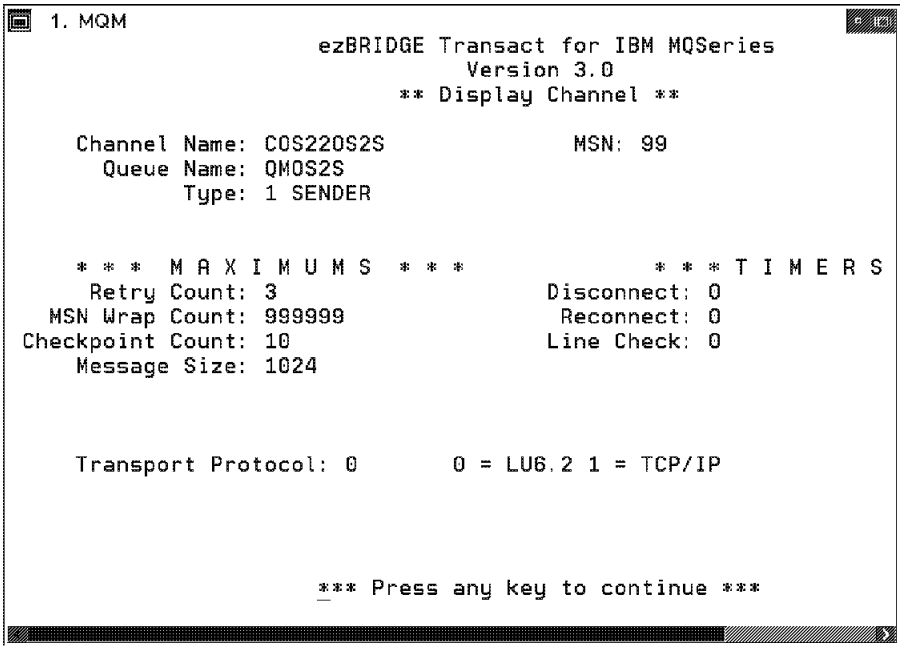

Figure 76. Define Sender Channel from OS22 (1 of 2)

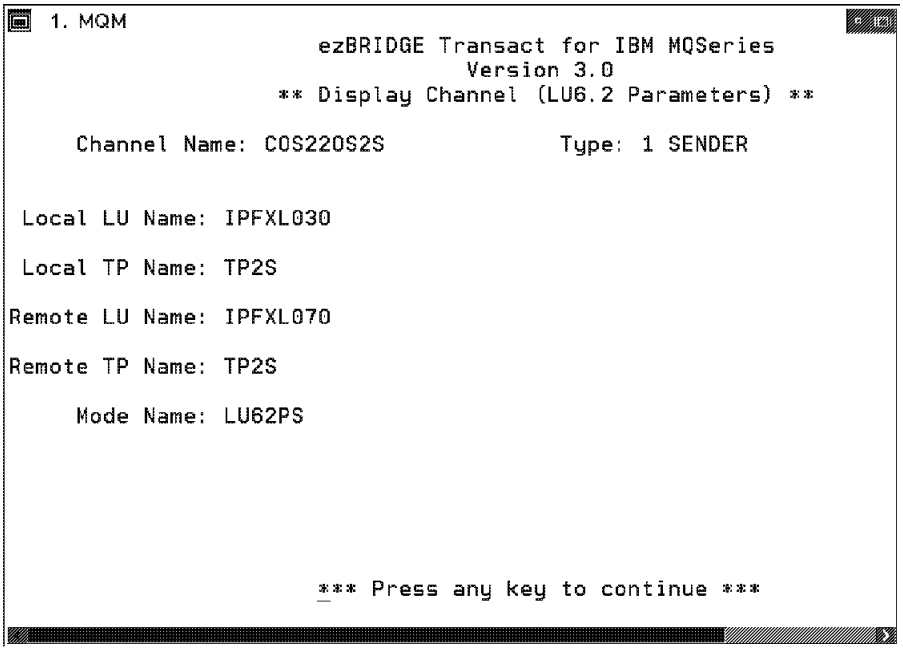

Figure 77. Define Sender Channel from OS22 (2 of 2)

Figure 78 on page 92 and Figure 79 on page 92 show the channel definitions for system OS22's receiver message channel called **COS2SOS22**. This channel is

started automatically by the sender MCA on the primary OS/2 system in our network configuration. It receives messages sent from the OS2S system. These messages are put on one of the two local queues FROMOS2S or FROMVSE. If the message header information specifies a queue which has not been defined here, the message is placed on the Dead Letter Queue.

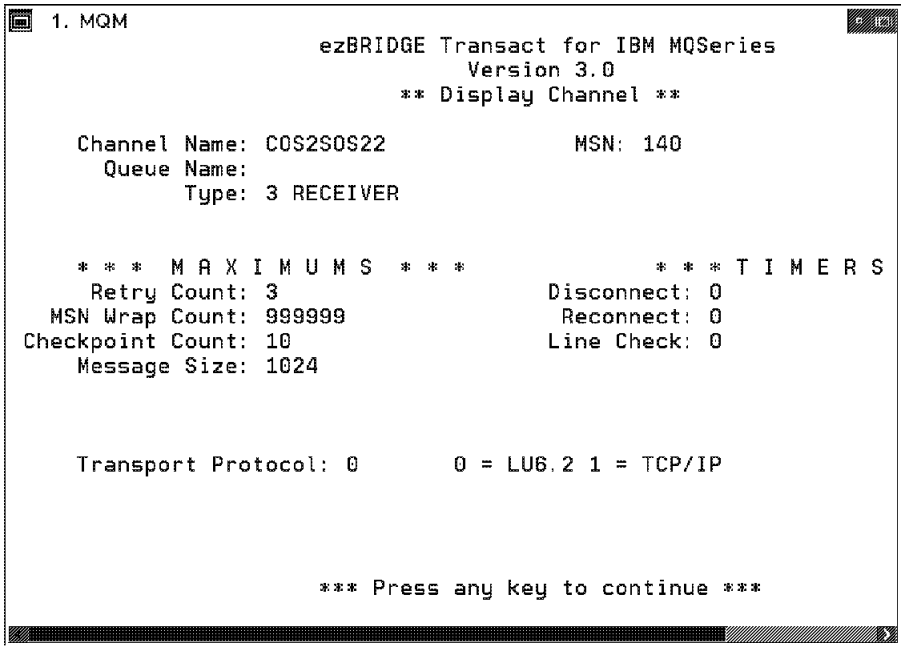

Figure 78. Define Receiver Channel from OS2S (1 of 2)

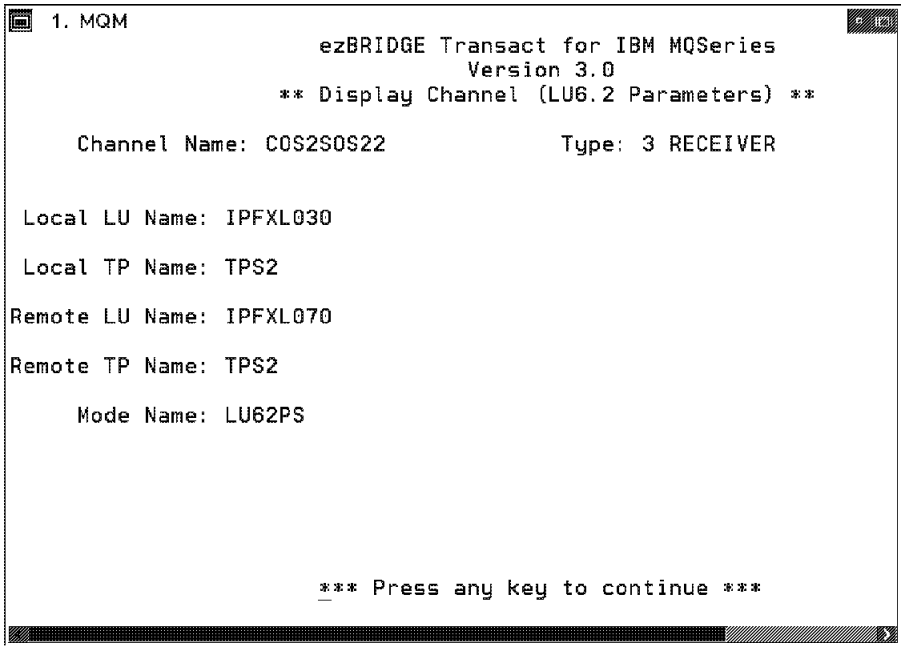

Figure 79. Define Receiver Channel from OS2S (2 of 2)

Chapter 6. MQI Applications for OS/2 and VSE/ESA

The programs used to test our scenario were those distributed with ezBRIDGE. The source code for these programs is distributed with the ezBRIDGE products. They are meant to be examples of how to code the MQI calls and can be used to verify your installation of ezBRIDGE. Here is a short description of the programs we used to test our scenario.

On the CICS/VSE host platform we used the distributed installation verification COBOL program called *'TTPTST2'*. This program can be invoked with the *'TST2'* transaction. It can be used to PUT and GET messages to and from ezBRIDGE queues.

To put 10 messages onto queue **TOOS2S**, for example, invoke TST2 this way:

- **TST2 PUT 10 TOOS2S**

On the OS/2 platform we used the distributed C language program called '**ZMQWRITE**'. The following command will write 10 messages of the five characters 'abcde' to queue **TOVSE** located on queue manager named **QMOS22**:

- zmqwrite QMOS22#TOVSE 10 5 abcde

Note that the queue manager and queue name are case sensitive. In our scenario, both names were defined in uppercase to MQM.

Here are some other special considerations for our VSE/ESA and OS/2 ezBRIDGE environments:

- Applications written for our scenario require programming skills in two languages, COBOL for the CICS/VSE environment and C for the PC environment. At the time of writing this document, neither of the two ezBRIDGE platforms used in our scenario provided facilities for message data conversion from one platform to another. Conversion of message data is the responsibility of the application program.
- The VSE/ESA implementation of message queue supports **triggers** which allow the automatic initiation of a program upon receipt of a message in a particular queue. Only pseudo-triggering is supported by ezBRIDGE on OS/2. With pseudo-triggering, an application must issue the message queue 'GET' function and then wait until a message arrives or the WaitInterval expires before regaining control. Thus OS/2 programs which read from Transact queues must be started externally.

Part 3. MQI Client/Server Implementation between VSE/ESA and AIX

Chapter 7. ezBRIDGE on VSE/ESA to ezBRIDGE on AIX Overview

The next four chapters describe the definitions required to implement the message queuing environment described in Figure 80 on page 100. IBM's Message Queue Interface (MQI) protocol is implemented by different products depending on the operating system platform, as described in 1.2, "The MQSeries Product Family" on page 5. The following chapters document the implementation of MQI solutions between AIX and VSE/ESA using:

- ezBRIDGE on AIX
- ezBRIDGE on VSE/ESA

7.1 ezBRIDGE on AIX

ezBRIDGE Transact on AIX/6000 for IBM MQSeries is a member of the IBM MQSeries family of products which allows AIX applications to exchange messages with other applications using the MQI. These other applications may reside on the same or on a different AIX system, or any other platform which is supporting the MQI.

Using MQI, AIX application programs can write messages on queues which either reside on the local file system or any other RISC System/6000 disk directory reachable over the network.

The queues are defined in a way which makes their location transparent to the application programs.

On AIX the MQI is built around the standard C language function call interface. The MQI functions are provided in the form of an object library.

Messages between different ezBRIDGE systems are handled by the **Message Channel Agent (MCA)**, a set of programs which implement a special protocol, the **Message Channel Protocol (MPC)**. On AIX the MCA uses either SNA LU 6.2 or TCP/IP protocols for communicating to other ezBRIDGE systems.

The message queues are managed and administered by the **Message Queue Management (MQM)** component of ezBRIDGE on AIX. The user (or administrator) interacts with MQM via a menu-driven program called ***MQM***.

7.1.1 Client/Server Support

ezBRIDGE on AIX provides for an ideal *client/server* environment based on MQI techniques. One AIX machine, the server, can provide queue management functions for a large number of ezBRIDGE on AIX clients.

In such a *distributed* MQI environment ezBRIDGE components and related MQI applications may reside on different LAN-connected systems and share a common file system.

Consider, for example, a Token-Ring LAN with several IBM RISC System/6000 systems attached:

- The RISC System/6000 running the MCA is called the Transact Communications *Server* (usually one per LAN). This server can act as a

gateway to other LANs or to other ezBRIDGE systems on other platforms, for example ezBRIDGE on VSE/ESA.

- The MQI services provided by the server are available to all applications in this network, that is all application programs using MQI functions, that is *clients* on the connected RISC System/6000 AIX systems.
- Queues may reside on any disk directory ′visible′ on the LAN.

The distributed RISC System/6000 environment described above with one Communications Server and one file system, is viewed as a single ezBRIDGE system or one *domain*.

The other extreme is one IBM RISC System/6000 which contains all ezBRIDGE components. Such an AIX system is called a *stand-alone* RISC System/6000 ezBRIDGE configuration.

7.2 Interoperation between ezBRIDGE on VSE/ESA and ezBRIDGE on AIX

ezBRIDGE on AIX may act in both roles, the server and stand-alone version, as a *gateway* to ezBRIDGE systems on other platforms.

To enable the *gateway* functions, the AIX system requires:

- an appropriate network attachment supporting IBM′s SNA LU 6.2 protocol, for example an IBM Token-Ring adapter and
- AIX SNA Services/6000 to provide for the communication software. If TCP/IP communication protocols are used no additional software is required.

Some of the differences between the AIX and VSE/ESA implementations of ezBRIDGE which are noteworthy in our scenario are:

- Applications distributed over these two platforms require programming skills in two languages, COBOL for the CICS/VSE environment and C for the AIX environment.
- The RISC System/6000 stores data in **ASCII** format. VSE/ESA uses **EBCDIC**. At the time of writing this document, neither of these two ezBRIDGE implementations provided facilities for message data conversion from one platform to another. Conversion of message data is the responsibility of the application program.
- The VSE/ESA implementation supports **triggers** which allow the automatic initiation of a program upon receipt of a message in a particular queue. Only pseudo-triggering is supported by ezBRIDGE on AIX. With pseudo-triggering, an application must issue the message queue ′GET′ function and then wait until a message arrives or the ′WaitInterval′ expires before regaining control. Thus AIX programs which read from ezBRIDGE queues must be started externally.
- The AIX implementation supports SNA LU 6.2 as well as TCP/IP protocols, whereas ezBRIDGE on VSE/ESA supports SNA LU 6.2 only.

Chapter 8. ezBRIDGE on VSE/ESA to ezBRIDGE on AIX Test Environment

This chapter describes the hardware and software used in our project and illustrates the MQI scenario between AIX and VSE/ESA.

Our environment consisted of:

- a VSE/ESA host running ezBRIDGEon VSE/ESA and
- an IBM RISC System/6000 workstation running ezBRIDGE on AIX.

The AIX workstation running ezBRIDGE and the ezBRIDGE VSE/ESA host are connected to each other via an IBM Token-Ring LAN running SNA protocols. The network is illustrated in Figure 80 on page 100.

8.1 Hardware

- An IBM 9221 Model 150 as the host with:
  - 64MB main memory
  - Token-Ring connection via channel attached IBM 3172
- An RISC System/6000 Model 320 workstation running ezBRIDGE on AIX

8.2 Software

- VM/ESA R2.1 with GCS and ACF/VTAM V3.4.1 in the IBM 9221
- VSE/ESA Version 1.3.3 guest under VM in the IBM 9221 with
  - ACF/VTAM V3.4.0
  - CICS/VSE V2.2
  - ezBRIDGE Transact on VSE/ESA for IBM MQSeries R3.0
- Interconnect Controller Program (ICP) Version 3.2 in the IBM 3172 Model 3
- AIX/6000 V3.2.5 running in the RISC System/6000 workstation with
  - AIX SNA Services/6000 V1.2
  - ezBRIDGE Transact on AIX/6000* for IBM MQSeries R3.0

8.3 Network Configuration

Figure 80 on page 100 illustrates the network described above including the MAC-addresses used in the appropriate VTAM, AIX SNA Services/6000, and 3172 ICP definitions.

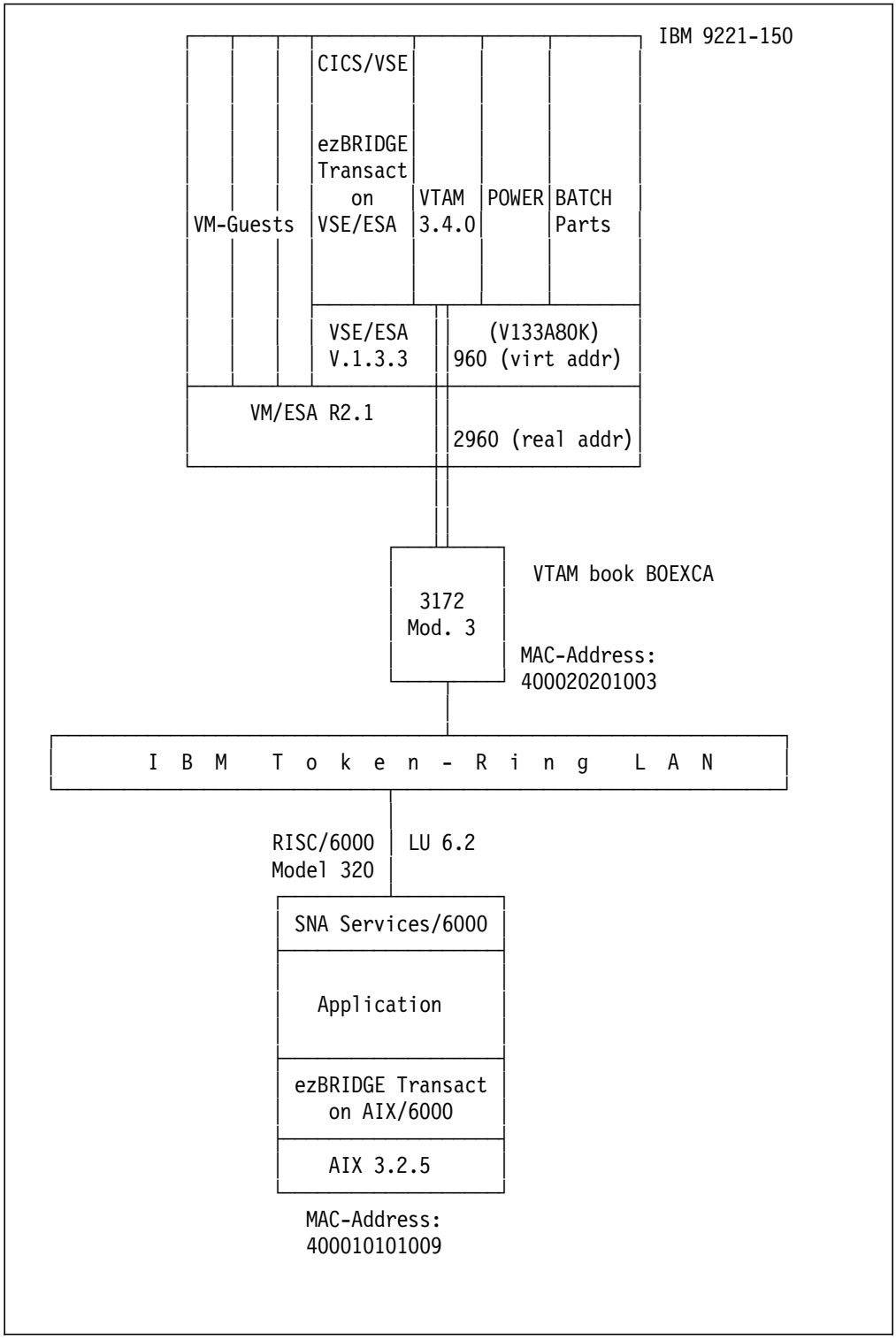

Figure 80. ezBRIDGE on VSE/ESA to ezBRIDGE on AIX MQI Network Diagram

8.4 ezBRIDGE on VSE/ESA to ezBRIDGE on AIX MQI Scenario

Figure 81 on page 102 illustrates our distributed MQI scenario from the message queuing point of view, that is it shows the interrelations of message queues, message channels and how the systems are connected to each other. This diagram should be referenced while studying the sample definitions provided in chapters 9 to 11.

The logic of the message flow between the AIX and VSE/ESA systems corresponds exactly to the message flows explained in the MQI scenarios shown in 3.4, “ezBRIDGE on VSE/ESA to ezBRIDGE on OS/2 MQI Examples” on page 13.

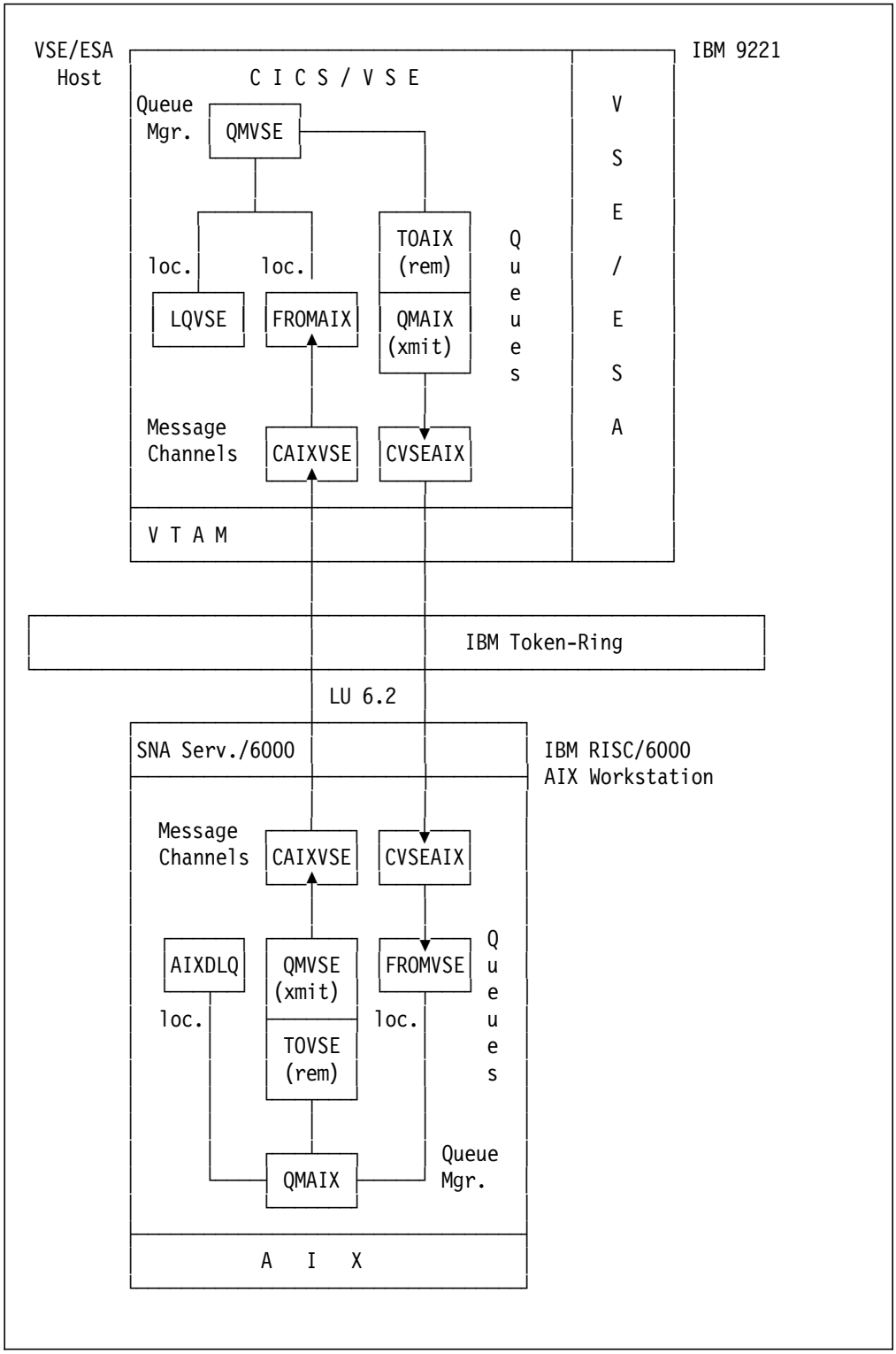

Figure 81. MQI-Communication between AIX and VSE/ESA

Chapter 9. Connecting the AIX Workstation to the Host

This chapter describes the implementation of the SNA LU 6.2 connection between VSE/VTAM and AIX SNA Services/6000, which is required for the communication between ezBRIDGE on VSE/ESA and ezBRIDGE on AIX. From all tasks we performed this turned out to be the most complex one due to:

- the number of components involved
- the lack of experience and documentation for this type of communication.

The implementation and customization of the connection consists of three parts which in turn involve a number of steps:

1. The definitions required for the host which consist of customizing
   - ACF/VTAM
   - CICS/VSE

   These definitions are provided in 9.2, “VSE/ESA Host Customization” on page 104.
2. The definitions required for RISC/6000 which consist of customizing
   - the Token-Ring adapter of the RISC System/6000
   - AIX SNA Services/6000

   These definitions are provided in 9.3, “AIX Customization” on page 109.
3. The operating instructions for starting and controlling the connection.

   These instructions are provided in 9.4, “Operational Hints” on page 126.

A summary of the customization process is given in 9.5, “VSE/ESA to AIX Connection Summary” on page 127.

9.1 Distributed MQI Support

ezBRIDGE on AIX supports distributed message queuing in two ways:

- Using a single queue manager.

  Here two application programs can connect to the same queue manager and exchange information through local queues managed by this queue manager.
- Using multiple queue managers.

  This allows application programs to write to remote queues and the local queue manager forwards the message(s) to the appropriate remote queue manager.

For the application program this process is transparent, that is it does not see any difference whether the queue is local or remote.

9.1.1 Network Protocol Considerations

When two systems communicate, they need to agree on a set of rules they will use to interpret the data they exchange. These rules are known as *network protocols* and they are defined in what is called a network architecture.

ezBRIDGE on AIX communicates with ezBRIDGE on other operating system platforms either via TCP/IP or SNA protocols. Which of the two protocols is used depends almost entirely on the type of the remote system. Non-UNIX platforms, such as VSE/ESA use SNA. UNIX-based platforms such as AIX will most likely use TCP/IP, although AIX also supports SNA protocols through AIX SNA Services/6000.

For the connection between ezBRIDGE systems on VSE/ESA and AIX the SNA LU 6.2 protocol is used.

9.1.2 Using SNA Communication

Since ezBRIDGE on VSE/ESA is a CICS/VSE application it uses CICS/VSE ISC functions to communicate with other ezBRIDGE systems.

For the SNA LU 6.2 connection required between ezBRIDGE on VSE/ESA and ezBRIDGE on VSE/ESA, both sides have to be customized appropriately as described in the subsequent chapters.

Any SNA communication to and from a RISC System/6000 requires that AIX SNA Services/6000 is installed on AIX.

9.2 VSE/ESA Host Customization

This chapter describes how to define the RISC System/6000 to VSE/VTAM, and to connect ezBRIDGE on AIX to ezBRIDGE on VSE/ESA.

For all other host definitions refer to Chapter 4, “ezBRIDGE on VSE/ESA Implementation” on page 19.

VSE/ESA customization involves three steps:

1. **Network attachment** to connect (in our configuration) the IBM 9221 to the IBM Token-Ring LAN (see Figure 80 on page 100). These definitions are already described in Chapter 4, “ezBRIDGE on VSE/ESA Implementation” on page 19.
2. **ACF/VTAM definitions** to provide for SNA LU6.2 communications.
3. **CICS/VSE definitions** to enable communication to and from AIX SNA Services/6000.

9.2.1 ACF/VTAM Customization

The RISC System/6000 is defined to ACF/VTAM via:

- a VTAM switched major node and
- an independent LU6.2 logical unit.

9.2.1.1 VSE/VTAM SWNET Major Node

A SWNET major node is created to define the physical and logical units to which VTAM may communicate through the 3172 XCA major node.

Figure 82 on page 106 shows the definitions we used:

Note the following parameters:

- ***IDBLK*** is 071. These are the first three hexadecimal digits of AIX SNA Services **Control Point** profile's attribute '**xid_node_id**'.
- ***IDNUM*** is E0009. These are the last five hexadecimal digits of AIX SNA Services **Control Point** profile's attribute '**xid_node_id**'.
- ***PATH*** The last six bytes of the DIALNO operand contain RISC System/6000 server's ALTERNATE TOKEN RING address.
- ***LOCADDR=0*** implies independent logical unit definition.
- ***DLOGMODE=LU62PS*** is the name of the log mode entry which describes the session parameters for the APPC conversations between CICS/VSE and ezBRIDGE on AIX.

  The log mode name **LU62PS** must match the mode name of an AIX SNA Services **LU6.2 Mode** profile, which must be referenced in the AIX SNA Services Connection Profile mode list. See Figure 12 on page 26 for details of the log mode entry.

In addition, the **LU** name **must** be the same as the ezBRIDGE queue manager name on AIX, *QMAIX* in our case.

```
*
*  3172 RELATED SWITCHED MAJOR NODE FOR  RS/6000
*
BOEXCASW       VBUILD TYPE=SWNET,MAXGRP=20,MAXNO=20
*
*  PU AND LU DEFINITION FOR RS/6000
*
IPFCPX09 PU    ADDR=03,           STATION ADDRESS FOR RS/6000          X
               LANSW=YES,         LAN capable                          X
               IDBLK=071,         IDENTIFICATION BLOCK                 X
               IDNUM=E0009,       IDNUM SET BY ITSC CONVENTION         X
               DISCNT=NO,         VTAM does not hang up                X
               ISTATUS=ACTIVE,PACING=0,VPACING=0,    NO PACING         X
               PUTYPE=2,          SNA cluster controller               X
               MAXDATA=521,         FROM 265 TO 2057                   X
               MAXOUT=7,            RECOMMENDED BY DOC                 X
               MAXPATH=1,           ADDED FROM LAB DEFN                X
               LANACK=(01.0,1),   LAN ACKNOWLEDGEMENT VALUES           X
               LANINACT=02.0,     timer for inactive link station      X
               LANCON=(05.0,1),   LAN TIMER AND RETRY COUNT            X
               LANSDWDW=(2,1),    SEND WINDOW & WINDOW STEP            X
               LANRESP=(02.0,2),  TIMER FOR CONNECTED STATE            X
               SAPADDR=4          service access point address
*
*   PATH STATEMENT FOR DIAL OUT
*
          PATH  DIALNO=0204400010101009
*
*   LU DEFINITION FOR RS/6000
*
QMAIX     LU    LOCADDR=0,DLOGMOD=LU62PS,            (APPC LU)        X
               ISTATUS=ACTIVE,MODETAB=CICSIPMT
```

Figure 82. ezBRIDGE on AIX VSE/VTAM Switched Major Node for 3172

9.2.2 CICS/VSE Customization

This chapter covers the definitions required in CICS/VSE to enable AIX SNA Services/6000 to communicate with CICS/VSE. The following CICS/VSE definitions are common for both, ezBRIDGE on AIX and ezBRIDGE on OS/2, and are described in4.3, “CICS/VSE Customization” on page 27:

- System Initialization Table (DFHSIT)
- Terminal Control Program (DFHTCP)
- Destination Control Table (DFHDCT)
- File Control Table (DFHFCT)

We used the ′**CEDA DEF**′ transaction to define the following resources to CICS:

- Connections
- Sessions

9.2.2.1 Connection Definition

Figure 83 shows the connection we defined for ezBRIDGE on AIX to ezBRIDGE on VSE/ESA communication.

```
OBJECT CHARACTERISTICS
 CEDA  VIew
  Connection     : AIXM
  Group          : MQIAIX
 CONNECTION IDENTIFIERS
  Netname        : QMAIX
  INDsys         :
 REMOTE ATTRIBUTES
  REMOTESystem   :
  REMOTEName     :
 CONNECTION PROPERTIES
  ACcessmethod   : Vtam              Vtam | IRc | INdirect
  Protocol       : Appc              Appc | Lu61
  SInglesess     : No                No | Yes
  Datastream     : User              User | 3270 | SCs | STrfield | Lms
  RECordformat   : U                 U | Vb
 OPERATIONAL PROPERTIES
  AUtoconnect    : No                No | Yes | All
  INService      : Yes               Yes | No
 SECURITY
  SEcurityname   :
  ATtachsec      : Local             Local | Identify | Verify
  Bindpassword   :                   PASSWORD NOT SPECIFIED



PF 1 HELP       3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 83. Connection for ezBRIDGE on AIX

The key parameters are:

- **CONNECTION** specifies the name of the connection.
- **NETNAME** specifies the name of the independent LU. It must match the:
  - Local LU NAME in the AIX SNA Services/6000 LU 6.2 Local LU Profile definition as shown in Figure 90 on page 117
  - Name of independent LU defined to VTAM as shown in Figure 82 on page 106.
- **SInglesess** specifies 'NO' in order to use APPC parallel sessions.
- **AUtoconnect=No** allows for manual control when the connection between CICS/VSE and the RISC System/6000 system is activated. This requires a '*CEMT SET CONN(sysid) ACQ*' (sysid is *AIXM* in our case) transaction to activate the connection. '**SInglesess**' must be set to '**No**' to provide parallel session (independent LU) support.

9.2.2.2 Session Definition

Figure 84 shows the definition we used to define the logical links and session characteristics between CICS/VSE and the independent LU for ezBRIDGE on AIX, QMAIX.

In the definition '**AUtoconnect=Yes**' should be specified. This will bind only the contention winners when the connection is started ('AUtoconnect=All' would cause CICS/VSE to bind all sessions).

The **MAximum** parameter is specified as 'XXX,YYY', where 'XXX' is the maximum number of sessions, and 'YYY' is the minimum number of contention winners. YYY should be greater than 0 and less than XXX. This will reserve some sessions for contention winners on the AIX side. The values will be negotiated when the connection is started, but they should be consistent with what is specified in the corresponding AIX SNA Services LU 6.2 Mode profile (see Figure 92 on page 119).

```
 OBJECT CHARACTERISTICS
  CEDA  View
   Sessions       : AIXM
   Group          : MQIAIX
  SESSION IDENTIFIERS
   Connection     : AIXM
   SESSName       :
   NETnameq       :
   MOdename       : LU62PS
  SESSION PROPERTIES
   Protocol       : Appc              Appc | Lu61
   MAximum        : 00008 , 00004     0-32767
   RECEIVEPfx     :
   RECEIVECount   : No                No | 1-999
   SENDPfx        :
   SENDCount      : No                No | 1-999
   SENDSize       : 00256             1-30720
   RECEIVESize    : 00256             1-30720
  OPERATOR DEFAULTS
   OPERId         :
   OPERPriority   : 000               0-255
   OPERRsl        : 0                                                 0-24,...
   OPERSecurity   : 1                                                 1-64,...
   USERId         :
  SESSION USAGES
   Transaction    :
   SESSPriority   : 100               0-255
  OPERATIONAL PROPERTIES
   Autoconnect    : Yes               No | Yes | All
   INservice      :                   No | Yes
   Buildchain     : Yes               Yes | No
   USERArealen    : 000               0-255
   IOarealen      : 00000 , 00000     0-32767
   RELreq         : No                No | Yes
   Discreq        : Yes               No | Yes
   NEPclass       : 000               0-255
  RECOVERY
   RECOvoption    : Sysdefault        Sysdefault | None

PF 1 HELP       3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 84. Session Definition for ezBRIDGE on AIX

The key parameters are:

- **SESSIONS.** Specifies the name of the session.
- **CONNECTION.** Defines the name of the connection associated with this session. It must match the CONNECTION name in CICS/VSE′s CONNECTION definition shown in Figure 83 on page 107.
- **MODENAME.** The logmode entry name for the independent LU named *QMAIX*. It must match the MODEENT name in the VTAM MODETAB (see Figure 12 on page 26).
- **AUTOCONNECT.** YES allows CICS to automatically establish a session with the session partner *QMAIX*.

All CICS/VSE resources were defined in group *MQIAIX*. After definition we used the **CEDA** transaction to activate and make them permanent in the CICS startup list:

- ***CEDA INSTALL GROUP(MQIAIX)***

  To activate them.

- ***CEDA ADD GROUP(MQIAIX) LIST(VSELIST)***

  To make them permanent in the list which is used when CICS starts.

9.3 AIX Customization

Before starting this task verify that AIX SNA Services/6000 is installed on your system. AIX SNA Services/6000 consists of two LPPs:

- **sna.lu0** for LU 0 support
- **sna.sna** for LU 6.2 support

AIX customization for ezBRIDGE on AIX involves the following tasks:

1. Customize the Token-Ring adapter of the RISC/6000
2. Customize AIX SNA Services/6000

9.3.1 Token-Ring Adapter Customization

For the IBM Token-Ring LAN connection to VSE/ESA, AIX must define the hardware address of its Token-Ring adapter. We use SMIT to change the address to an ALTERNATE TOKEN RING address, that is a locally administered address:

1. Enter ′**smitty devices**′.
2. Move the cursor to the ′**Communication**′ field and press ′Enter′.
3. Select the ′**Token Ring Adapter**′ field.
4. In the Token Ring Adapter panel select ′**Adapter**′.
5. Select field ′**Change / Show Characteristics of a Token Ring Adapter**′ and move the cursor to the appropriate item and press ′Enter′ to obtain the following panel:

```
              Change / Show Characteristics of a Token Ring Adapter


Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    ′Entry Fields′
  Token Ring Adapter                                tok0
  Description                                       Token-Ring High-Perfor>
  Status                                            Available
  Location                                          00-04
  Receive data transfer OFFSET                      ′92′                    +#
  TRANSMIT queue size                               ′30′                    +#
  RECEIVE queue size                                ′30′                    +#
  STATUS BLOCK queue size                           ′10′                    +#
  RING speed                                        16                      +
  Receive ATTENTION MAC frame                       no                      +
  Receive BEACON MAC frame                          no                      +
  Enable ALTERNATE TOKEN RING address               yes                     +
  ALTERNATE TOKEN RING address                      ′0x400010101009′
  Apply change to DATABASE only                     no                      +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 85. RISC System/6000 Token-Ring Adapter Customization

6. Enable ALTERNATE TOKEN RING address if it is not enabled.
7. Move the cursor to field ′**ALTERNATE TOKEN RING address**′ and enter an address (400010101009 in our example).

   The address must match the value of the DIALNO parameter of the PATH statement in VTAM′s PU definition (see Figure 82 on page 106).
8. Press ′Enter′ to make the changes effective.

If you intend to have a host connection via NCP and your NCP Release is lower than 6, the Token-Ring hardware address must contain only numeric characters (0-9).

9.3.2 AIX SNA Services/6000 Customization

Before we customized AIX SNA Services/6000 on our system we applied the following PTFs on top of our version 1.2 of the LPP:

1. U426472
2. U426473
3. U428048

The customization of AIX SNA Services/6000 can be divided into three parts.

1. The customization of profiles describing its **physical** characteristics, that is:
   a. the SNA Token-Ring Attachment Profile
   b. the SNA Token-Ring Logical Link Profile
   c. the SNA Token-Ring Physical Link Profile
2. The customization of profiles describing its **logical** characteristics, that is:

a. the SNA LU6.2 Logical Connection Profile

b. the SNA LU6.2 Local LU Profile

c. the SNA LU6.2 Mode List Profile

d. the SNA LU6.2 Mode Profile

e. the SNA LU6.2 TPN List Profile

f. the SNA LU6.2 TPN Profile

g. the SNA LU6.2 RTPN List Profile

h. the SNA LU6.2 RTPN Profile

3. The customization of:

   a. the SNA Node Profile

   b. the SNA Control Point Profile

We used SMIT to create or change the appropriate SNA profiles in the following way:

1. Enter ′**smitty sna**′.
2. Move the cursor to the ′**Configure SNA Profiles**′ field and press ′Enter′.
3. Select the ′**Advanced SNA Configuration**′ field.
4. Select the appropriate components or items in the following panels to create or modify their profiles.

9.3.2.1 Change Profiles for SNA Physical Units

1. **Token-Ring Attachment Profile**

   This profile defines the physical characteristics of the communications environment. It includes data link control, transmission media, and adapter assignment. Once started, an attachment provides connectivity and SNA data transport across the physical link.

   The profile in Figure 86 on page 112 shows the profile we used to connect the RISC/6000 to CICS/VSE via the Token-Ring adapter.

```
                Change / Show SNA Token Ring Attachment Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

'TOP'                                                    'Entry Fields'
  CURRENT profile name                                  LU62EZ
  NEW PROFILE name                                     ''
  CONTROL POINT profile name                           'LU62EZ'
  LOGICAL LINK profile name                            'LU62EZ'
  PHYSICAL LINK profile name                           'LU62EZ'
  STOP ATTACHMENT on inactivity?                        no                      +
     If yes, inactivity TIMEOUT (0-10 minutes)         '5'                      #
  RESTART on deactivation?                              yes                     +
  LU address REGISTRATION?                              no                      +
     If yes, LU address REGISTRATION PROFILE name      'LDEFAULT'               +
  CALL type                                             call
     If listen,
        AUTO-LISTEN?                                    yes                     +
        MINIMUM SAP address (hex 04-ec)                '04'
        MAXIMUM SAP address (hex 04-ec)                'EC'
     If call, ACCESS ROUTING                            link_address
        If link_name, REMOTE LINK name                 ''
        If link_address,
           Remote LINK address                         '400020201003'
           Remote SAP address (hex 04-ec)              '04'
'BOTTOM'

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
```

Figure 86. LU62EZ Attachment Profile

We configured the attachment profile with **CALL type=call** and physical **Remote LINK address**, which is the Token-Ring MAC address of the IBM 3172 (refer to Figure 80 on page 100).

This configuration does not allow for connections to multiple systems. In **call** mode, only one single Token-Ring address may be used to communicate with the Token-Ring adapter on the RISC System/6000 system, since AIX/SNA Services only contacts that particular address after having started the attachment profile. Other devices with different Token-Ring addresses cannot establish communication with the RISC System/6000 system using this attachment.

If **listen** mode is specified, AIX SNA Services can accept Token-Ring connections from any number of other devices at the same time. The 'listen' attachment requires that the non-AIX communication partner initiates connection to the RISC/6000; this allows multiple MQI systems on different platforms, to connect to ezBRIDGE on AIX at the same time.

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the set of characteristics described in this profile.
- **CONTROL POINT profile name.** This field provides the name of the Control Point Profile that defines the node ID of the physical unit associated with this attachment (refer to 9.3.2.3, "Change Profiles for SNA Nodes" on page 124).

- **LOGICAL LINK profile name.** This field provides the name of the Logical Link Profile that defines the characteristics of the data link protocol that implements the network (refer to Figure 87 on page 113).
- **PHYSICAL LINK profile name.** This field provides the name of the Physical Link Profile that defines the characteristics of the physical port for the network (refer to Figure 88 on page 114).
- **Remote LINK address.** When *CALL type* is *call* and *ACCESS ROUTING* is *link_address*, this field specifies the network address of the remote station (3172 MAC address in our example see Figure 80 on page 100).

2. **Token-Ring Data Link Control Logical Link**

   This profile is required by the Attachment Profile and contains information regarding the data link level. In a valid configuration, at least one Logical Link must exist.

   Figure 87 shows the Logical Link Profile we used.

```
                Change / Show SNA Token Ring Logical Link Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

'TOP'                                                 'Entry Fields'
  CURRENT profile name                               LU62EZ
  NEW PROFILE name                                  ''
  TRANSMIT window count (1-127)                     '10'                      #
  DYNAMIC window increment (1-127)                  '1'                       #
  RETRANSMIT count (1-30)                           '8'                       #
  RECEIVE window count (1-127)                      '127'                     #
  RING ACCESS priority                               0                       +
  RETRY limit                                       '20'                      #
  DROP LINK on inactivity?                           yes
  INACTIVITY timeout (1-120 seconds)                '48'                      #
  RESPONSE timeout (1-40, 500 msec intervals)       '2'                       #
  ACKNOWLEDGE timeout (1-40, 500 msec intervals)    '1'                       #
  FORCE DISCONNECT timeout (1-600 seconds)          '120'                     #
  DEFINITION of maximum I-FIELD size                 system_defined          +
     If user_defined, max. I-FIELD SIZE (265-30729) '30729'                    #
  TRACE link?                                        no                      +
     If yes, TRACE SIZE                              short                   +
'BOTTOM'

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 87. LU62EZ Token-Ring Logical DLC Profile

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the set of characteristics described in this profile. The profile name appears in the Attachment profile (see Figure 86 on page 112) for the attachment that uses this profile.

3. **Token-Ring Data Link Control Physical Link**

   This profile is required by the Attachment Profile and contains information regarding the physical level. For a valid configuration, at least one Logical Link must exist.

   Figure 88 shows the Physical Link Profile we used.

```
                Change / Show SNA Token Ring Physical Link Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      'Entry Fields'
  CURRENT profile name                               LU62EZ
  NEW PROFILE name                                  ''
  DATALINK device name                              'tok0'
  LOCAL LINK name                                   ''
  Maximum number of LOGICAL LINKS (1-255)           '32'                    #
  Local SAP address (hex 04-ec)                     '04'                    X









F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 88. LU62EZ Token-Ring Physical DLC Profile

The key parameters are:

a. **PROFILE name.** The system uses this name to refer to the set of characteristics that you describe in this profile. The profile name also appears in the Attachment Profile (see Figure 86 on page 112) for the attachment that uses this profile.

b. **DATALINK device name.** This field contains the name that the local system (AIX) uses for the Token-Ring attachment.

9.3.2.2 Change Profiles for SNA LU6.2 Units

1. **SNA LU6.2 Logical Connection Profile**

   To use a connection, you must describe its characteristics to AIX SNA Services/6000 Connection Profile.

   This is the 'key' profile for defining ezBRIDGE on AIX within AIX SNA Services/6000 and describes ezBRIDGE's SNA resources. It defines a **LOCAL** and **REMOTE LU** pair, an **ATTACHMENT**, and a set of session characteristics.

   This profile (named 'LU62EZ' in our example) will be referenced in the ezBRIDGE's *Channel Definitions*, see Figure 114 on page 144, and Figure 116 on page 146.

   Figure 89 shows the Connection Profile we used.

```
                Change / Show SNA LU6.2 Logical Connection Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

'TOP'                                                    'Entry Fields'
  CURRENT profile name                                  LU62EZ
  NEW PROFILE name                                     ''
  ATTACHMENT profile name                              'LU62EZ'
  LOCAL LU profile name                                'LU62EZ'
  NETWORK name                                         'DEIBMIPF'
  STOP CONNECTION on inactivity?                        no                    +
     If yes, TIMEOUT (0-10 minutes)                    '0'                    #
  REMOTE LU name                                       'CICSSA22'
  SECURITY Accepted                                     none                  +
     If conversation or already_verified,
     CONVERSATION SECURITY ACCESS LIST profile         'CONVDEFAULT'
        (If no name entered, /etc/passwd used)
  REMOTE TPN LIST profile name                         'RDEFAULT'
  MODE LIST profile name                               'LU62EZ'
  INTERFACE type                                        extended
     If extended, SESSION CONCURRENCY                   parallel
  Node VERIFICATION?                                    no                    +
'BOTTOM'

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Figure 89. LU62EZ Connection Profile

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the remote LU associated with the profile and to refer to the set of characteristics described.
- **ATTACHMENT profile name.** This field provides the name of the Attachment Profile that describes the characteristics of the attachment to the remote LU (see Figure 86 on page 112).
- **LOCAL LU profile name.** This field provides the name of the Local LU Profile that defines the characteristics of the local LU (see Figure 90 on page 117).

- **NETWORK name.** This field provides the network name associated with remote LU for this connection. This name, combined with the remote LU name forms a fully qualified remote LU name (*'DEIBMIPF.CICSSA22'* in our case).
- **REMOTE LU name.** This field provides the name of the remote LU representing the other half of this connection. The name must match the VTAM APPLID name of CICS/VSE.
- **MODE LIST profile name.** This field specifies the name of the Mode List Profile that contains the Mode Profile applying to this connection (see Figure 91 on page 118).
- **SESSION CONCURRENCY.** Must specify 'parallel'.

At least one LU6.2 Connection Profile must exist for each ezBRIDGE on AIX system that wants to use a remote connection via SNA protocols.

Note: The terminology used in AIX SNA Services/6000 differs from the one used on other platforms, ACF/VTAM and CM/2, for example.

2. **SNA LU6.2 Local LU Profile**

   This profile is required by the SNA LU6.2 Logical Connection Profile. It defines the characteristics of the local LU, that is ezBRIDGE on AIX. At least one Local LU Profile is required for each ezBRIDGE on AIX system that uses an SNA connection.

   Figure 90 shows the Local LU Profile we used.

```
                    Change / Show SNA LU6.2 Local LU Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      'Entry Fields'
  CURRENT profile name                               LU62EZ
  NEW PROFILE name                                  ''
  TPN LIST profile name                             'VTPN'              +
  NETWORK name                                      'DEIBMIPF'
  Local LU NAME                                     'QMAIX'
  INDEPENDENT LU?                                    yes                    +
     If no,
        Local LU ADDRESS (1-255)                    '1'                     #
        SSCP ID                                     ''




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 90. LU62EZ Local Logical Unit

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the local LU associated with the profile and to refer to the set of characteristics described.
- **TPN LIST profile name.** This field specifies the name of the TPN List Profile that lists the TPN Profiles that can use this LU (see Figure 93 on page 120). This applies only to LUs of type 6.2.
- **NETWORK name.** This field provides the network name associated with the local LU. This name, combined with the local LU name forms a fully qualified local LU name.

  If you are attaching to an existing network, use the name of that network.
- **LOCAL LU name.** This field provides the name of the local LU. The name must match the VTAM LU definition (see Figure 82 on page 106) and CICS/VSE connection definition (see *Netname* in Figure 83 on page 107).

An **SSCP ID** is not specified for *independent* type 6.2 LUs, because they are not controlled by an SNA host.

3. **SNA LU6.2 Mode List Profile**

   This profile is required by the SNA LU6.2 Connection Profile. It contains a list of Mode Profiles that describe the characteristics of LU6.2 sessions associated with the connection.

   Figure 91 shows our Mode List Profile.

```
                        Add SNA LU6.2 Mode List Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

'TOP'                                                    'Entry Fields'
* PROFILE name                                          'LU62EZ'
  Add profile names to list:
     Name  1                                           'LU62PS'
     Name  2                                           ''                          +
     Name  3                                           ''                          +
     Name  4                                           ''                          +
     Name  5                                           ''                          +
     Name  6                                           ''                          +
     Name  7                                           ''                          +
     Name  8                                           ''                          +
     Name  9                                           ''                          +
     Name 10                                           ''                          +
     Name 11                                           ''                          +
'MORE...53'

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 91. LU62EZ Mode List Profile

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the set of characteristics described in this profile.
- **Add profile name to list.** These fields contain the names of Mode Profiles available for use on the session which is associated with this list.

An entry which is identical to the Log Mode specified in the VSE/ESA ACF/VTAM definition (see Figure 82 on page 106) should be the first one on this list; this entry is the default entry.

4. **SNA LU6.2 Mode Profile**

   This profile is required by the SNA LU 6.2 Mode List Profile. It names an SNA mode, and defines the attributes of that mode. The mode defines the characteristics of sessions between the local and remote LUs specified in the corresponding Connection Profile. The name of the defined mode can be the same as the name of the Mode Profile that defines it (however, this is not required).

   At least one (default) Mode Profile must exist, specifying the name of the mode table entry defined in the VTAM independent LU (usually specified in the DLOGMOD parameter of the LU definition, see Figure 82 on page 106). The names of the host VTAM mode table entry (see Figure 12 on page 26) and the AIX SNA Services mode name (**LU62PS** in our case) **must** be the same. Additional Mode Profiles may be defined as required.

   Figure 92 shows our Mode Profile.

```
                    Change / Show SNA LU6.2 Mode Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    'Entry Fields'
  CURRENT profile name                             LU62PS
  NEW PROFILE name                                 ''
  MODE name                                        'LU62PS'
  Maximum number of SESSIONS (1-999)               '8'                    #
  Minimum contention WINNERS (0-499)               '4'                    #
  Minimum contention LOSERS (0-500)                '4'                    #
  Auto ACTIVATIONS limit (0-500)                   '0'                    #
  RECEIVE pacing (0-63)                            '3'                    #
  SEND pacing (0-63)                               '3'                    #
  Maximum RU SIZE (256,288,...,3840)               '2816'                 #
  RECOVERY level                                    no_reconnect          +




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 92. LU62PS Mode Profile

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the set of characteristics described in this profile.
- **MODE name.** This field associates the set of session characteristics that are described in this profile with a mode name used by SNA. This mode name **must** be defined consistently across the network.

5. **SNA LU 6.2 Transaction Program Name (TPN) List Profile**

   This profile is required by the SNA LU 6.2 Local LU Profile. It consists of a list of TPN Profiles.

   Figure 93 shows the TPN List Profile we used.

```
                        Add SNA LU6.2 TPN List Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

'TOP'                                                    'Entry Fields'
* PROFILE name                                         'VTPN'
  Add profile names to list:
     Name  1                                           'VTPN'
     Name  2                                           ''                          +
     Name  3                                           ''                          +
     Name  4                                           ''                          +
     Name  5                                           ''                          +
     Name  6                                           ''                          +
     Name  7                                           ''                          +
     Name  8                                           ''                          +
     Name  9                                           ''                          +
     Name 10                                           ''                          +
     Name 11                                           ''                          +
'MORE...53'

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 93. VTPN TPN List

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the set of characteristics described in this profile.
- **Add profile names to list.** These fields contain the names of TPN profiles available for use on the session associated with this list (refer to Figure 90 on page 117).

This profile must include the TPN profiles used by the ezBRIDGE's *sender* and *receiver* channels (see Figure 114 on page 144 and Figure 116 on page 146).

6. **SNA LU6.2 Transaction Program Name (TPN) Profile**

   This profile is required by the TPN List Profile. It describes the characteristics of the transaction programs that may be started by a remote SNA host. The name of this profile must match the *Template Transaction Program Name Profile* attribute of ezBRIDGE's *sender* and *receiver* channels (see Figure 114 on page 144 and Figure 116 on page 146).

   Figure 94 shows our TPN Profile.

```
                    Change / Show SNA LU6.2 TPN Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

'TOP'                                                 'Entry Fields'
  CURRENT profile name                               VTPN
  NEW PROFILE name                                  ''
  Transaction program name is in HEXADECIMAL?        no                      +
  TRANSACTION program name                          ''
  PIP data?                                          no                      +
     If yes, SUBFIELDS (0-99)                       '0'                      #
  CONVERSATION type                                  mapped                  +
  RECOVERY level                                     no_reconnect            +
  SYNC level                                         either                  +
  Full PATH to TPN executable                       '/usr/lpp/sna'
  MULTIPLE INSTANCES supported?                      no                      +
  User ID                                           '0'                      #
  SERVER synonym name                               ''
  RESTART action                                     once                    +
  COMMUNICATION type                                 signals                 +
     If IPC, communication IPC queue key            '0'                      #
  Standard INPUT file/device                        '/dev/null'
  Standard OUTPUT file/device                       '/dev/console'
  Standard ERROR file/device                        '/dev/console'
  SECURITY Required                                  none                    +
     If access,
        RESOURCE SECURITY ACCESS LIST profile       'RSRCDEFAULT'
        (If no name entered, /etc/passwd used)
'BOTTOM'

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 94. VTPN TPN Profile

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the set of characteristics described in this profile.
- **Full PATH to TPN executable.** Defines the path to AIX SNA Services/6000 within AIX.

7. **SNA LU 6.2 Remote Transaction Program Name (RTPN) List Profile**

   This profile is required by the SNA LU 6.2 Logical Connection Profile. At least one RTPN List Profile is required which in turn has to contain the name of at least one RTPN Profile which describes transaction **MQ01**.

   Figure 95 shows the RTPN List Profile we used.

```
                    Change SNA LU 6.2 RTPN List Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

'TOP'                                                    'Entry Fields'
  CURRENT profile name                                  RDEFAULT
  NEW PROFILE name                                     ''
  DELETE profile names from list (F4 to list)                              +
  Add profile names to list:
     Name  1                                           'MQ01'
     Name  2                                           ''                  +
     Name  3                                           ''                  +
     Name  4                                           ''                  +
     Name  5                                           ''                  +
     Name  6                                           ''                  +
     Name  7                                           ''                  +
     Name  8                                           ''                  +
     Name  9                                           ''                  +
'MORE...54'

F1=Help           F2=Refresh        F3=Cancel         F4=List
F5=Reset          F6=Command        F7=Edit           F8=Image
F9=Shell          F10=Exit          Enter=Do
```

Figure 95. RDEFAULT REMOTETPNLIST Profile

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the list of profile names specified in this profile.
- **Add profile names to list.** These fields contain the names of transaction program name profiles available for use on the session associated with this remote transaction program name list.

8. **SNA LU 6.2 Remote Transaction Program Name (RTPN) Profile**

   This profile is required by the SNA LU 6.2 RTPN List Profile. It describes the characteristics of remote transaction programs representing remote MCAs.

   Figure 96 shows our RTPN Profile.

```
                    Change / Show SNA LU 6.2 RTPN Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    'Entry Fields'
  CURRENT profile name                             MQ01
  NEW PROFILE name                                 ''
  RTPN name is in HEXADECIMAL?                     no                    +
  RTPN name                                        'MQ01'
  PIP data?                                        no                    +
  CONVERSATION type                                mapped                +
  RECOVERY level                                   no_reconnect          +
  SYNC level                                       none                  +

F1=Help          F2=Refresh       F3=Cancel        F4=List
F5=Reset         F6=Command       F7=Edit          F8=Image
F9=Shell         F10=Exit         Enter=Do
```

Figure 96. MQ01 Remote TPN Profile

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the set of characteristics described in this profile.
- **RTPN name.** The name of a remote MCA.

This profile is used by ezBRIDGE *sender* or *requester* channels to initiate a conversation between themselves and *receiver* or *server* channels on another ezBRIDGE system.

9.3.2.3 Change Profiles for SNA Nodes

1. **SNA Node Profile**

   This profile is required to start the SNA subsystem. The name of the SNA profile **must** be *'sna'*.

   Figure 97 shows our SNA Node Profile.

```
                        Change / Show SNA Node Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      'Entry Fields'
  CURRENT profile name                                 sna
  NEW PROFILE name                                    ''
  Total active open CONNECTIONS (1-5000)              '200'                    #
  Total SESSIONS (1-5000)                             '200'                    #
  Total CONVERSATIONS (1-5000)                        '200'                    #
  SERVER synonym name                                 'sna'
  RESTART action                                       once                   +
  Perform ERROR LOGGING?                               no                     +
  Standard INPUT file/device                          '/dev/null'
  Standard OUTPUT file/device                         '/dev/console'
  Standard ERROR file/device                          '/dev/console'




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 97. SNA Profile

2. **SNA Control Point Profile**

   This profile is required by the SNA Token-Ring Attachment Profile. It supplies PU information to configure the workstation as a node in a particular network.

   At least one Control Point Profile is required in a valid configuration. If more Control Point Profiles exist, and are active at the same time on the same or different networks, then XID data and control point (CP) name (if needed) must be unique within the network.

Figure 98 shows our SNA Control Point Profile Definitions.

```
                    Change / Show SNA Control Point Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        'Entry Fields'
  CURRENT profile name                                  LU62EZ
  NEW PROFILE name                                     ''
  XID node ID                                          '071E0009'
  NETWORK name                                         'DEIBMIPF'
  CONTROL POINT name                                   'IPFCPX11'









F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 98. LU62EZ Control Point Profile

The key parameters are:

- **PROFILE name.** The system uses this name to refer to the set of characteristics described in this profile.
- **XID node ID** is a hexadecimal value that provides the node ID of the physical unit, and is exchanged with the remote physical unit when a connection is first established.

  It consists of two fields:

  – The first three hexadecimal digits must correspond to the *IDBLK=* parameter in the VTAM PU definition statement.

  – The last five hexadecimal digits must correspond to the *IDNUM=* parameter in the VTAM PU definition statement.

  Refer to Figure 82 on page 106 to cross-check these values.

9.4 Operational Hints

This section provides some hints related to AIX SNA Services/6000 and how to connect to VSE/VTAM and CICS/VSE.

SMIT can be used to control AIX SNA Services/6000 resources using the following commands:

- To start AIX SNA resources:
  - **startsrc -s sna**

    Starts the *sna* subsystem.
  - **startsrc -t attachment -o *LU62EZ***

    Starts the attachment *LU62EZ*.
  - **startsrc -t connection -o *LU62EZ***

    Starts the connection *LU62EZ*. You can also start the connection from the CICS/VSE side by entering

    CEMT SET CONN(*AIXM*) ACQ

    at the CICS/VSE console.
- To display the status of AIX SNA resources you can use the following commands:
  - **lssrc -l -s sna**

    Shows the status of *sna* resources.
  - **lssrc -l -t attachment -o *LU62EZ***

    Displays the status of attachment *LU62EZ*.
  - **lssrc -l -t connection -o *LU62EZ***

    Displays the status of connection *LU62EZ*.
- To stop AIX SNA resources:
  - **stopsrc -t attachment -o *LU62EZ***

    Stops attachment *LU62EZ*.
  - **stopsrc -t connection -o *LU62EZ***

    Stops connection *LU62EZ*.
  - **stopsrc -s sna**

    Stops the *sna* subsystem.

9.5 VSE/ESA to AIX Connection Summary

The tables below list the items to be customized both for the VSE/ESA host and ezBRIDGE on AIX together with a reference to the corresponding topic in this or other chapters.

1. **VSE/ESA Host Customization Summary**

VSE/ESA Host	Items to be customized	Details to be found in
Network Attachment	IOCDS and VM considerations	4.2.1.1, “IBM 3172 Definition in IBM 9221 IOCDS” on page 20
	Attach the IBM 3172 to VSE-Machine	4.2.2.1, “Attach the IBM 3172 to the VSE/ESA Machine” on page 20
	Define the IBM 3172 in VSE/ESA	4.2.2.2, “Define the IBM 3172 in VSE/ESA” on page 21
	Customize the IBM 3172	4.2.3, “IBM 3172 Customization” on page 21
VSE/VTAM	VTAM XCA Major Node	4.2.4.1, “VTAM XCA Major Node” on page 23
	VTAM Switched Major Node	4.2.4.2, “VSE/VTAM Switched Major Node” on page 24
	VTAM Log Mode Entry	4.2.4.3, “VSE/VTAM Logmode Table Entry” on page 25
	VTAM Appl. Major Node	4.2.4.4, “VTAM Application Major Node” on page 26
CICS/VSE Resources	DFHSIT	4.3.1.1, “System Initialization Table (SIT)” on page 28
	DFHTCP	4.3.1.2, “Terminal Control Program (DFHTCP)” on page 29
	DFHDCT	4.3.1.3, “Destination Control Table (DFHDCT)” on page 30
	DFHFCT	4.3.1.4, “File Control Table (DFHFCT)” on page 30
	Connections	9.2.2.1, “Connection Definition” on page 107
	Sessions	9.2.2.2, “Session Definition” on page 108
	Programs	4.3.2.3, “Program and Transaction Definition” on page 36

Table 1. VSE/ESA Connection Customization Summary

2. **AIX Customization Summary**

AIX Server	Items to be customized	Details to be found in
AIX	*/etc/environment* file	Figure 99 on page 129
RISC/6000	TR-Adapter Hardware Address	9.3.1, “Token-Ring Adapter Customization” on page 109
SNA Services/6000	**Physical characteristics profiles**	9.3.2.1, “Change Profiles for SNA Physical Units” on page 111
	- Attachment Profile	Figure 86 on page 112
	- Token-Ring DLC Logical Link	Figure 87 on page 113
	- Token-Ring DLC Physical Link	Figure 88 on page 114
	Logical characteristics profiles	9.3.2.2, “Change Profiles for SNA LU6.2 Units” on page 115
	- LU6.2 Logical Connection	Figure 89 on page 115
	- LU6.2 Local Logical Unit	Figure 90 on page 117
	- LU6.2 Mode List	Figure 91 on page 118
	- LU6.2 Mode	Figure 92 on page 119
	- LU6.2 Transaction Program Name List	Figure 93 on page 120
	- LU6.2 Transaction Program Name	Figure 94 on page 121
	- LU6.2 Remote Transaction Program Name List	Figure 95 on page 122
	- LU6.2 Remote Transaction Program Name	Figure 96 on page 123
	Nodes	9.3.2.3, “Change Profiles for SNA Nodes” on page 124
	- System Network Architecture	Figure 97 on page 124
	- Control Point	Figure 98 on page 125

Table 2. AIX Connection Customization Summary

Chapter 10. ezBRIDGE on AIX Implementation

10.1 Overview

Installing ezBRIDGE on AIX on an IBM RISC System/6000 involves three steps:

1. Check/customize the AIX environment.
2. Install the ezBRIDGE on AIX Server.
3. Install the ezBRIDGE on AIX Client.

Only the basic steps are described in this document for ezBRIDGE on AIX installation. For details refer to *ezBRIDGE Transact on AIX/6000 for MQSeries User Manual, SC33-1143* and *Examples of Using MQSeries on S/390, RISC System/6000, AS/400 and PS/2, GG24-4326*.

10.1.1 AIX Environment Customization

Two items need to checked in the AIX environment and changed if necessary:

1. The default values of the environment variable *LANG*. As shipped, Transact assumes this value to be *En_US* (United States English).
2. In the *PATH* statement */usr/bin* needs to be specified.

Verify these items and change them if necessary in the */etc/environment* file:

```
PATH=<current path>:/usr/sbin
LANG=En_US
```

Figure 99. Verify AIX Environment for ezBRIDGE on AIX

If you made changes you must re-boot AIX to activate the environment variables.

Note: Some users may have *.profile* scripts that totally override environment variables set in */etc/environment*. In these cases the *.profile* for users who want to use the MQI provided by ezBRIDGE on AIX must be adjusted to include the above specified values.

It is recommended that users build their own PATH specification on top of the original */etc/environment* PATH specification (for example, PATH=$PATH:/home/...).

10.1.2 Installing the ezBRIDGE on AIX Server

As mentioned before there are two ways to install and use ezBRIDGE on AIX:

- As **stand-alone** system: Runs on a single IBM RISC System/6000. In such an environment, only application programs executing in the same RISC/6000 machine can use the facilities provided by the ezBRIDGE on AIX product.
- As **client/server** system: Multiple RISC/6000 systems can access the queues defined to one (the server) queue manager. The set of RISC/6000 systems

which access the same queues is called *domain*. The queue manager and configuration directory reside on one system, the server.

The installation of the *stand-alone* and *server* part of an ezBRIDGE on AIX configuration in a *client/server* environment is almost the same.

To install an ezBRIDGE on AIX client/server configuration follow the list of tasks provided below. For a *stand-alone* ezBRIDGE on AIX system the last step exporting directories) is not required.

We installed an ezBRIDGE on AIX *stand-alone* system, following the steps described in *ezBRIDGE Transact on AIX/6000 for MQSeries User Manual, SC33-1143*.

1. Log into AIX system as ′root′ user.
2. Insert ezBRIDGE on AIX distribution media, and execute the *tar* command to install the software into the AIX system:

   tar -xvf /dev/rmt0
3. Change directories to the product install directory:

   cd /usr/lpp/mqi/install
4. Execute the installation script *ezbinstall*.

   ezbinstall

   This script adds symbolic links to the system and creates the directories required.
5. Verify that */usr/sbin* exists in your PATH environment variable:

   echo $PATH

   See 10.1.1, “AIX Environment Customization” on page 129.
6. Define a directory to contain the ezBRIDGE on AIX queue manager files.

   When configuring the Queue Manager, a directory path for all its configuration files and queues must be specified (see Figure 102 on page 134, ′Config Path′ attribute).

   It is recommended to use a separate file system for this directory for the following reasons:

   - To simplify backup procedure: all queue manager definitions and the queues including application data will be stored in this directory.
   - To simplify client configuration: all client systems can access this directory via NFS ′mount′.

     In our sample configuration we used ezBRIDGE′s */var/mqi* directory.

   The next three steps are only required when ezBRIDGE on AIX communicates with the other queue managers, either over TCP/IP or SNA.
7. The message channel agent maintenance demon (**mcamd**) requires two TCP/IP sockets for communication, two sockets are needed for it to operate the UNIX** domain and the INTERNET socket. A UNIX domain socket is created by the demon in */var/mqi*. In order to create the INTERNET socket, a new service called *ezbmcamd* must be registered in file */etc/services*. This is done by adding an appropriate entry for *ezbmcamd* in the */etc/services* file as shown in Figure 100 on page 131.

```
    .
    .
    .
#
# ezBRIDGE Definition
#
ezbmcamd     2300/tcp       # Message Channel Agent demon
    .
    .
    .
```

Figure 100. ezBRIDGE on AIX MCA Demon

2300 is the TCP/IP port number assigned to the ezBRIDGE on AIX MCA; any other *unique* number greater than 1024 could also be specified.

The new definition is activated by entering:

inetimp

8. This step is directly related to the previous step. The MCA demon (*mcamd*) is only required for distributed queue processing.

   Start the *mcamd* demon by entering:

   mcamd &

   Starting the demon with the *'&'* parameter starts it in the background, thereby releasing the AIX session where the command was executed.

   Note: The *mcamd* process needs to be running at all times for ezBRIDGE programs to operate.

9. Configure SNA Services/6000 to support SNA LU 6.2 communications to ezBRIDGE on VSE/ESA (see 9.3.2, "AIX SNA Services/6000 Customization" on page 110).

10. Run *mqm* to create the queue manager configuration. See 10.2, "ezBRIDGE on AIX Customization" on page 133 for details.

11. This step is required only if you are running an ezBRIDGE on AIX *client/server* configuration, as described above.

    Export the following directories via NFS:

    - /var/mqi
    - /usr/lpp/mqi
    - The directory created in step 6, and defined in the queue manager's *Config Path* field.

    See 10.1.3, "Installing ezBRIDGE on AIX Client System" on page 132.

At this point the ezBRIDGE on AIX *stand-alone* and/or *server* system can be further connected to other ezBRIDGE systems running on other platforms.

10.1.3 Installing ezBRIDGE on AIX Client System

The *client* systems in an ezBRIDGE on AIX client/server environment are part of the server's domain and can run MQI applications. MQI applications running on client systems obtain all MQI resources (queues and message channels) from the server. Consequently the configuration of a client system does not require installation of any ezBRIDGE code. The only action required for the client is to get access to the ezBRIDGE product code and the server's queue manager data via NFS (Network File System):

1. This first step corresponds to the last step of the ezBRIDGE on AIX server installation described before.

   The directories used by ezBRIDGE on AIX system must be made available for NFS export. Add the following lines to the */etc/export* file on the server system:

```
    .
    .
    .
#
# ezBRIDGE Directories
#
/var/mqi -access=userid1:userid2:...
/usr/lpp/mqi -access=userid1:userid2:...
'directory defined in step 6 of server installation' -access=userid1:userid2:..
    .
    .
    .
```

Figure 101. Prepare ezBRIDGE on AIX Server for NFS Export

   The *-access* option is not mandatory, but should be used if you want to restrict access to these directories to a set of users (or AIX groups).

2. To make these directories accessible for ezBRIDGE on AIX clients immediately after adding them to */etc/export*, they have to be 'explicitly' exported by the server as shown by the following commands:

   exportfs /var/mqi

   exportfs /usr/lpp/mqi

   exportfs *'directory from step 6'*

   These commands need to be executed only once. The next time NFS is started it will automatically export these directories.

3. An ezBRIDGE on AIX client who wants to run MQI applications needs to 'import' the directories which were exported by the server via NFS 'mount' commands. This involves two steps.

   a. Define the directories to be mounted:

      mkdir /var/mqi

      mkdir /usr/lpp/mqi

      mkdir *'directory from step 6 of server installation'*

   b. Mount the exported server directories 'over' the directories defined in the previous step:

mount /var/mqi

mount /usr/lpp/mqi

mount *'directory from step 6 of server installation'*

4. Establish the symbolic links required to run ezBRIDGE on AIX, by executing following commands on client system:

   cd /usr/lpp/mqi/install

   install_links

5. Make sure that */usr/sbin* exists in client system's PATH environment variable:

   echo $PATH

   See 10.1.1, "AIX Environment Customization" on page 129.

10.2 ezBRIDGE on AIX Customization

Configuration, object definitions, system operations and monitoring of ezBRIDGE on AIX is done by using the ezBRIDGE program **mqm**. This program is invoked by entering its name at the AIX prompt:

mqm

10.2.1 Configuring ezBRIDGE on AIX for Local Use

In this part we followed the ezBRIDGE on AIX Installation Verification Test (IVT) procedure described in the ezBRIDGE Transact on AIX/6000 for MQSeries User Manual, SC33-1143. The IVT uses one local queue and the sample programs **zmqwirte** and **zmqread** provided with ezBRIDGE on AIX. The sample programs are stored in directory */usr/lpp/mqi/bin*, which usually is not part of the PATH specification. Therefore, you need to position yourself at this directory for this test:

cd /usr/lpp/mqi/bin

When you invoke *mqm* the first time after ezBRIDGE on AIX product installation the *Message Queue Manager Definition* (see Figure 102 on page 134) panel is displayed.

The entries described below have to be made and the information has to be saved by pressing keys *Ctrl* and *W* (*mqm* uses this combination of keys for saving changes), before ezBRIDGE on AIX can be used for any further administrative activity or application access.

Figure 102 on page 134 shows our queue manager definitions.

```
                      ezBRIDGE Transact for IBM MQSeries
                                 Version 3.0
                              ** Queue Manager **

         Name: QMAIX
  Description: Q Manager on AIX
  Config Path: /var/mqi
Dead Letter Q: AIXDLQ
    Char. Set: 850
  Max Handles: 1
  Max Message: 4096
Max Poll time: 100
 MCA Hostname: aix320









<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 102. ezBRIDGE on AIX Message Queue Manager Configuration Screen

Note the following parameters:

- **Name:** Specifies the name of the local queue manager.
- **Config Path:** This field defines the directory where the configuration information and the queues will be stored (see step 6 of 10.1.2, “Installing the ezBRIDGE on AIX Server” on page 129).
- **Dead Letter Q:** Specifies the name of the queue used by ezBRIDGE on AIX for storing messages which cannot be delivered to their intended destination queue. Such messages will require operator action to recover.

The screen can be left either via <Ctrl-X>, to discard changes, or with <**Ctrl-W**>, to save changes.

Note: Any specified value can be changed at a later time via the ′Modify Queue Manager′ function. However, changes to this information should be avoided, as they may cause incompatibilities with existing applications and definitions for other queue managers connected to this queue manager.

After defining the Queue Manager perform the following steps:

1. At the system prompt type **mqm**, to enter the ′Main Menu′.
2. In the ′Main Menu′ select ′*Configuration*′ by typing:

   1
3. Select ′*Create Queue*′ by typing:

   3
4. This brings you to the ′Define Queue Name′ panel. Figure 103 on page 135 shows the definition of the name for our local queue.

```
                      ezBRIDGE Transact for IBM MQSeries
                                Version 3.0
                           ** Define Queue Name **

Queue Type: L      L=Local,   R=Remote,   AQ=Alias Queue
                   AM=Alias Queue Manager
                   AR=Alias Reply Queue

Name: ANYQ









<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 103. ezBRIDGE on AIX Define Queue Name Panel

Fill in data in the screen. Key parameters are:

- **Queue Type:** The type determines whether this is a local or remote queue. An alias queue is a 'logical queue', that is refers to an existing local or remote queue with another name.
- **Name:** This is the name of the queue being defined.

5. Press < **Ctrl-W** > to save the data.
6. The 'Create Local Queue' screen appears. Figure 104 shows our definitions for queue *ANYQ*.

```
                      ezBRIDGE Transact for IBM MQSeries
                                Version 3.0
                           ** Create Local Queue **

       Name:  ANYQ
Description:  any local Q
      Usage:  0   0 = Normal,    1 = Transmission
  File Name:  test

Max Queue Depth:  100
    Max Message:  4096









<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 104. ezBRIDGE on AIX Create Local Queue Panel

The key parameters are:

- **Usage:** *'Normal'* means that the queue can be used by local applications for reading and writing messages or to receive inbound messages.
- **File Name:** Identifies two files, named *Test.que* and *Test.qul*, which will be created in the directory specified in the queue manager configuration.
- **Max Queue Depth:** Specifies the maximum number of messages allowed on this queue.
- **Max Message:** The maximum length of messages processed on this queue.

7. Fill in appropriate data and press < **Ctrl-W** > to save them.
8. Press < **Ctrl-X** > to return to the 'Configuration Menu'. Select option **6** 'Display Queue' to verify your queue definition.
9. The panel 'Select Queue To Display' appears:

```
                      ezBRIDGE Transact for IBM MQSeries
                                Version 3.0
                        ** Select Queue To Display **


AIXDLQ                                              LOCAL
ANYQ                                                LOCAL









J       -  Down              K       -  Up              <return>  -  Select
CTRL-F  -  PgDn              CTRL-B  -  PgUp            CTRL-X    -  Exit
```

Figure 105. ezBRIDGE on AIX 'Select Queue To Display' Panel

Use **J** and **K** to select the queue and press <**Enter**> .

10. A 'Display Local Queue' screen will display the queue parameters just entered. After verification press the <**Enter**> key.
11. Press two times <**Ctrl-X**> to return to the 'Main Menu'.

    Now, you are ready for the local IVT.
12. From the 'Main Menu', select the 'Monitoring' option by entering **3**. The 'Monitor Menu' screen shown in Figure 106 on page 137 appears.

```
                      ezBRIDGE Transact for IBM MQSeries
                                 Version 3.0
                              ** Monitor Menu **



                                 Enter Choice: 1

                                 1.  Monitor Queue

                                 2.  Monitor Channel









<return> - Select Option <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Go to previous menu
```

Figure 106. ezBRIDGE on AIX Monitor Queue Selection Menu

13. From the 'Monitor Menu', select the 'Monitor Queue' option by entering **1**.
14. The 'Monitor Queues' panel in Figure 107 shows the values for our local queue named *ANYQ* we defined in the previous steps. There are no messages currently stored on this queue which is indicated by **DEPTH** being equal to **0**.

```
                      ezBRIDGE Transact for IBM MQSeries             14:26:03
                                 Version 3.0
                              ** Monitor Queues **


Queue                                          Type     USERS LWRIT DEPTH G P
================================================================================
AIXDLQ                                         LOCAL    00000 00000 00000 A A
ANYQ                                           LOCAL    00000 00000 00000 A A









J       -  Down              K       -  Up              <return>  -  Select
CTRL-F  -  PgDn              CTRL-B  -  PgUp            CTRL-X    -  Exit
```

Figure 107. ezBRIDGE on AIX Monitor Queues Result Panel

15. Now we are ready to run the IVT. For this purpose we move to another AIX window and change the current directory to */usr/lpp/mqi/bin* by entering:

    cd /usr/lpp/mqi/bin

16. Use the test program *zmqwrite* to send some messages to the local queue ANYQ. At the system prompt, type:

    zmqwrite QMAIX#ANYQ 10 1000 " any text "

17. Successful completion of the zmqwrite-command can be checked on the previous panel (Figure 107). The number of messages on queue ANYQ (DEPTH) must now be '10'.

18. We also run the test program *zmqread* from another AIX window to read some messages from our local queue ANYQ. At the system prompt, type:

    zmqread QMAIX#ANYQ 10

    The messages are displayed on the screen as they are read.

19. Successful completion of the zmqread-command can be checked on the panel in Figure 107 on page 137. The number of messages on queue ANYQ (DEPTH) must now be '0' again.

20. After having completed the IVT we leave mqm by pressing <**Ctrl-X**> .

10.2.2 Configuring ezBRIDGE on AIX for Communication to ezBRIDGE on VSE/ESA

For communication to ezBRIDGE on VSE/ESA using SNA LU 6.2 protocols the following preparation is required for ezBRIDGE on AIX:

1. Install and customize AIX SNA Services/6000 to provide for SNA LU 6.2 connections for AIX. This is described in Chapter 9, "Connecting the AIX Workstation to the Host" on page 103.
2. Define ezBRIDGE message channels between ezBRIDGE on AIX and ezBRIDGE on VSE/ESA.
3. Define remote and transmission queues for exchanging messages between ezBRIDGE on AIX and ezBRIDGE on VSE/ESA.

As explained in 1.1, "IBM's Message Queue Interface: Basic Concepts" on page 3, transmission queues provide the 'link' between queue managers on different systems. They **must** exist before any remote queue is defined. ezBRIDGE checks the existence of the transmission queue specification during remote queue definition.

The queue and message channel definitions shown in subsequent chapters refer to our ezBRIDGE on AIX to ezBRIDGE on VSE/ESA MQI scenario illustrated in Figure 81 on page 102.

10.2.2.1 Configuring Transmission Queues

Defining a transmission queue is very similar to local queue definition. To create transmission queue '*QMVSE*' for sending messages to ezBRIDGE on VSE/ESA we performed the following steps as shown in Figure 108 on page 139 to Figure 109 on page 139:

1. Invoke **mqm** and select '*Configuration*'.
2. In the 'Configuration Menu' select option '*Create Queue*'.
3. Fill in the fields as displayed below.

```
                        ezBRIDGE Transact for IBM MQSeries
                                  Version 3.0
                              ** Define Queue Name **

Queue Type: L      L=Local,   R=Remote,   AQ=Alias Queue
                   AM=Alias Queue Manager
                   AR=Alias Reply Queue

Name: QMVSE









<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 108. Define Transmission Queue QMVSE

Key parameters are:

- **Queue Type:** 'L' needs to be specified since a transmission queue is a special case of a local queue.
- **Name:** This is the name of the queue being defined.

4. Press < **Ctrl-W** > to save the data.
5. The 'Create Local Queue' screen will appear:

```
                        ezBRIDGE Transact for IBM MQSeries
                                  Version 3.0
                              ** Create Local Queue **

       Name:  QMVSE
Description:  Transmission Queue to QMVSE
      Usage:  1   0 = Normal,    1 = Transmission
  File Name:  qmvse

Max Queue Depth:  10000
    Max Message:  1024

<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 109. Create Transmission Queue QMVSE

The key parameters are:

- **Usage:** ′1′ specifies a transmission queue. This queue cannot be used by local applications to read and write messages. It holds *outbound* messages destined for the queue manager *QMVSE* on ezBRIDGE on VSE/ESA.
- **File Name:** Two files, named *qmvse.que* and *qmvse.qul*, will be created in the directory of queue manager QMAIX.
- **Max Queue Depth:** Specifies the maximum number of messages allowed on this queue.
- **Max Message:** Specifies the maximum length of messages for this queue.

6. Enter the required data and press < **Ctrl-W** > to save it.
7. Press < **Ctrl-X** > to return to the ′Configuration Menu′.

Note: A transmission queue is always associated with a message channel, CAIXVSE in our case (see Figure 113 on page 143). The messages on a transmission queue are processed only by ezBRIDGE′s Message Channel Agent (MCA).

10.2.2.2 Configuring Remote Queues

If messages are to be sent to remote ezBRIDGE systems a remote queue must be defined. A remote queue is a logical name definition on the local system to identify a queue which physically resides on a remote system.

To define remote queue *TOVSE* we performed the following steps:

1. Go to the ′Define Queue Name′ menu using the ′**mqm**′ command as shown in 10.2.2.1, “Configuring Transmission Queues” on page 138.
2. Specify ′R′ for remote queue and ′TOVSE′ as its name:

```
                      ezBRIDGE Transact for IBM MQSeries
                                Version 3.0
                           ** Define Queue Name **

Queue Type: R      L=Local,   R=Remote,   AQ=Alias Queue
                   AM=Alias Queue Manager
                   AR=Alias Reply Queue

Name: TOVSE









<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 110. Define Remote Queue TOVSE

3. Press < **Ctrl-W** > to save the data.
4. The ′Create Remote Queue′ screen appears:

```
                        ezBRIDGE Transact for IBM MQSeries
                                   Version 3.0
                              ** Create Remote Queue **

       Name:  TOVSE
Description:  Remote Q


   Remote Queue Name:  FROMAIX
Remote Queue Manager:  QMVSE
 Transmit Queue Name:









<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 111. Create Remote Queue TOVSE

The key parameters are:

- **Remote Queue Name:** Identifies the remote queue on the *local* system.
- **Remote Queue Manager:** 'QMVSE' is the queue manager's name on ezBRIDGE on VSE/ESA on which TOVSE is defined as a local queue.
- **Transmit Queue Name:** QMVSE is TOVSE's associated transmission queue.

  This field can be left blank if the transmission queue name is the same as the remote queue manager name (this is true in our case).

5. Enter the required data and press < **Ctrl-W** > to save it.
6. Press < **Ctrl-X** > to return to the 'Configuration Menu'.

Note that a transmission queue must exist for a remote queue before a remote queue can be created.

We created another local queue named ***FROMVSE*** for messages coming from VSE/ESA. In ezBRIDGE on VSE/ESA this queue is known as remote queue **TOAIX**, see 4.4.3, "ezBRIDGE on VSE/ESA to ezBRIDGE on AIX Configuration" on page 49.

After we created all queues we need, we can display them:

```
                    ezBRIDGE Transact for IBM MQSeries
                              Version 3.0
                        ** Select Queue To Display **


AIXDLQ                                          LOCAL
ANYQ                                            LOCAL
FROMVSE                                         LOCAL
QMVSE                                           TRANSMIT
TOVSE                                           REMOTE









J       -  Down             K       -  Up           <return>  -  Select
CTRL-F  -  PgDn             CTRL-B  -  PgUp         CTRL-X    -  Exit
```

Figure 112. Display Existing Queues

10.2.2.3 Defining Message Channels

As described before a message channel is a uni-directonal point-to-point communications link between two ezBRIDGE systems, ezBRIDGE on VSE/ESA and ezBRIDGE on AIX in our example. This connection is implemented through a message channel definition, on on each side.

In our configuration we defined *receiver/sender* channels on both VSE/ESA and AIX. They represent the *server* and *client* parts of our distributed MQI environment. The *sender* channel on ezBRIDGE on AIX has a *receiver* partner on ezBRIDGE on VSE/ESA, and vice versa.

To define and use ezBRIDGE message channels in an AIX environment the following is required:

- An MCA demon (*mcamd*) must be running, see 10.1.2, “Installing the ezBRIDGE on AIX Server” on page 129 step 8.
- AIX SNA Services/6000 profiles must be configured to provide for the SNA LU 6.2 link required if the remote ezBRIDGE system is ezBRIDGE on VSE/ESA (see 9.3.2, “AIX SNA Services/6000 Customization” on page 110).

For an overview of the naming conventions used for message channel and message channel agent definitions refer to C.5, “Message Channel and MCA Summary” on page 163.

1. **Define Sender Channel CAIXVSE**

   CAIXVSE is the message channel for sending messages to ezBRIDGE on VSE/ESA and uses the transmission queue QMVSE defined in 10.2.2.1, “Configuring Transmission Queues” on page 138.

   To define sender channel *CAIXVSE* perform the following steps:

   a. Go to the ′Create Channel′ menu using the ′**mqm**′ command as shown in 10.2.2.1, “Configuring Transmission Queues” on page 138.

   b. Enter the data as displayed in the screen below:

```
                      ezBRIDGE Transact for IBM MQSeries
                                 Version 3.0
                             ** Create Channel **

     Channel Name: CAIXVSE                MSN: 1
       Queue Name: QMVSE
             Type: 1 1 = Sender 2 = Server 3 = Receiver  4 = Requester


     * * *  M A X I M U M S  * * *             * * * T I M E R S * * *
      Retry Count: 3                    Disconnect: 0
   MSN Wrap Count: 1000                  Reconnect: 0
 Checkpoint Count: 10                   Line Check: 0
     Message Size: 1024



     Transport Protocol: 0       0 = LU6.2 1 = TCP/IP



<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 113. Create Sender Channel CAIXVSE (1 of 2)

Key parameters are:

- **Channel Name:** Specifies the name of the message channel to be created.
- **MSN:** Initial value for the Message Sequence Number.
- **Queue Name:** Refers to our associated transmission queue ′QMVSE′, see 10.2.2.1, “Configuring Transmission Queues” on page 138.
- **Type:** ′1′ creates a sender channel
- **Transport Protocol:** ′**0**′ (zero) selects SNA LU 6.2.

   c. Press < **Ctrl-W** > to save the data entered.

   d. The ′Create Channel (LU6.2 Parameters)′ screen appears:

```
                    ezBRIDGE Transact for IBM MQSeries
                                Version 3.0
                  ** Create Channel (LU6.2 Parameters) **

     Channel Name: CAIXVSE              Type: 1 SENDER


     Device Name: /dev/sna          Connection Profile: LU62EZ

   Remote Transaction Program Name Profile: MQ01









<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 114. Create Sender Channel CAIXVSE (2 of 2)

The key parameters are:

- **Device Name:** Device name should always be */dev/sna*. This is a standard device definition in AIX SNA Services/6000.
- **Connection Profile:** Specifies the name of the *SNA LU6.2 Logical Connection Profile* that must be defined to the SNA Services to describe the connection between this and the remote queue manager (see Figure 89 on page 115).
- **Remote Transaction Program Name Profile:** Specifies the name of the *Remote Transaction Program Name Profile* defined to AIX SNA Services/6000 (see Figure 96 on page 123). This is the transaction program that represents the MCA on ezBRIDGE on VSE/ESA to handle incoming messages from ezBRIDGE on AIX.

  Transaction *MQ01* is supplied by ezBRIDGE on VSE/ESA.

e. Press <**Ctrl-W**> to save the data entered.

f. Press <**Ctrl-X**> to return to the 'Configuration Menu'.

2. **Define Receiver Channel CVSEAIX**

   CVSEAIX is the message channel for receiving messages from ezBRIDGE on VSE/ESA.

   The definition process is the same as for the sender channel CAIXVSE discussed in the previous step. The screens below show the definitions we made:

```
                      ezBRIDGE Transact for IBM MQSeries
                                 Version 3.0
                              ** Create Channel **

     Channel Name: CVSEAIX                MSN: 1
       Queue Name:
             Type: 3 1 = Sender 2 = Server 3 = Receiver  4 = Requester


     * * *  M A X I M U M S  * * *             * * * T I M E R S * * *
      Retry Count: 3                    Disconnect: 0
   MSN Wrap Count: 1000                  Reconnect: 0
 Checkpoint Count: 10                   Line Check: 0
     Message Size: 1024



     Transport Protocol: 0       0 = LU6.2 1 = TCP/IP



<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 115. Create Receiver Channel CVSEAIX (1 of 2)

```
                    ezBRIDGE Transact for IBM MQSeries
                                Version 3.0
                  ** Create Channel (LU6.2 Parameters) **

    Channel Name: CVSEAIX              Type: 3 RECEIVER


    Device Name: /dev/sna          Connection Profile: LU62EZ

 Template Transaction Program Name Profile: VTPN
                         Transaction Name: VTPN









<return> - FIELD EXIT    <esc>  - Discard Field Changes   CTRL-D - Erase Field
<BKSP>   - BACKSPACE     CTRL-X - Exit discarding change  CTRL-W - Save Change
```

Figure 116. Create Receiver Channel CVSEAIX (2 of 2)

The key parameters are:

- **Connection Profile:** Specifies the name of the *SNA LU 6.2 Logical Connection Profile* that must be defined in AIX SNA Services/6000 to describe the connection between this and the remote queue manager (see Figure 89 on page 115).

  Note: We are using the same connection as defined for the *sender* channel; it supports *parallel* sessions.

- **Template Transaction Program Name Profile:** Specifies the name of the *Transaction Program Name Profile* defined in AIX SNA Services/6000 (see Figure 94 on page 121).

- **Transaction Name:** 'VTPN' name identifies the transaction program, that is the MCA responsible for handling incoming messages from ezBRIDGE on VSE/ESA (see Figure 94 on page 121).

Chapter 11. MQI Applications for AIX and VSE/ESA

The programs used to test our scenario were those distributed with ezBRIDGE. The source code for these programs is distributed with the ezBRIDGE products. They are meant to be examples of how to code the MQI calls and can be used to verify your installation of ezBRIDGE. Here is a short description of the programs we used to test our scenario.

On the CICS/VSE host platform we used the distributed installation verification COBOL program called *TTPTST2*. This program can be invoked with the *TST2* transaction. It can be used to PUT and GET messages to and from ezBRIDGE queues.

On the AIX platform we used the distributed C language program called *zmqwrite* and *zmqread* to write and read messages to and from ezBRIDGE queues.

ezBRIDGE on AIX sample programs are stored in the */usr/lpp/mqi/bin* directory, which is usually not part of the PATH specification. Therefore, you would typically need to position yourself at this directory before executing these programs:

cd /usr/lpp/mqi/bin

While reading the following description please refer to Figure 81 on page 102.

Message transfer between ezBRIDGE systems on distributed platforms can be performed in a simple way by writing messages in *remote* queues.

11.1 Message Transfer from VSE/ESA to AIX

To put 10 messages on a queue called *TOAIX* on VSE/ESA, invoke TST2 this way:

- **TST2 PUT 10 TOAIX**

Note: Between *PUT* and *10* (number of messages) two blanks have to be inserted.

TOAIX is a remote queue on ezBRIDGE on VSE/ESA, and its intention is to route messages to queue *FROMVSE* on ezBRIDGE on AIX, which is a local queue for AIX.

If an SNA LU 6.2 connection between ezBRIDGE on VSE/ESA and ezBRIDGE on AIX is established, ten messages will arrive in queue *FROMVSE* as the result of the TST2 PUT described above.

Messages can be received via the ezBRIDGE on AIX supplied program *zmqread* by entering

- **zmqread QMAIX#FROMVSE 10**

  at the AIX command prompt.

Successful execution of *zmqread* displays messages together with their sequence numbers on the AIX window.

11.2 Message Transfer from AIX to VSE/ESA

The following command writes 10 messages consisting of the five characters ′abcde′ to remote queue *TOVSE* on AIX:

- **zmqwrite QMAIX#TOVSE 10 5 ″abcde″**

Queue *TOVSE* is defined as remote to ezBRIDGE on AIX′s queue manager for sending messages to ezBRIDGE on VSE/ESA′s local queue *FROMAIX*.

If the connection between ezBRIDGE on AIX and ezBRIDGE on VSE/ESA is established, then the ten messages will arrive in *FROMAIX* on VSE/ESA.

Messages can be received at the host via CICS/VSE MQI application program *TST2* invoked by the *TST2* transaction. On the CICS/VSE console enter:

- **TST2 GET 10 FROMAIX**

Note: The queue manager and queue names on AIX are case sensitive. In our MQI scenario, we always used uppercase characters to identify ezBRIDGE resources.

11.3 Message Transfer Considerations

Here are some considerations regarding our VSE/ESA and AIX ezBRIDGE environment:

- If a connection between ezBRIDGE on AIX and ezBRIDGE on VSE/ESA is not established, then messages are held in local transmission queues (*QMAIX* on VSE/ESA, or *QMVSE* on AIX). When the connection becomes operational, messages from AIX will be automatically transferred to VSE/ESA. On VSE/ESA an additional message must be ′PUT′ into the appropriate remote queue to trigger the transfer.
- The sender channel (CAIXVSE) on AIX must be activated, if the corresponding receiver channel (CAIXVSE) on VSE/ESA is in an IDLE state.
- Applications written for our scenario require programming skills in two languages, COBOL for the CICS/VSE and C for the AIX environment.
- At the time of writing this document, neither of the two ezBRIDGE platforms used in our scenario provided facilities for message data conversion from one platform to another (ASCII to EBCDIC and vice versa). Conversion of message data is the responsibility of the application program.
- The VSE/ESA implementation of message queue supports **triggers** which allow the automatic initiation of a program upon receipt of a message in a particular queue. Only pseudo-triggering is supported by ezBRIDGE on AIX. With pseudo-triggering, an application must issue the message queue ′GET′ function and then wait until a message arrives (if the queue is empty), or the WaitInterval expires before regaining control. Thus AIX programs which read from ezBRIDGE on AIX queues must be started externally.

Appendix A. ezBRIDGE on VSE/ESA Sample Definitions

A.1 VTAM Start List

```
* $$ JOB JNM=BOESTR04,DISP=D,PRI=3,                                    C
* $$ NTFY=YES,                                                         C
* $$ LDEST=*,                                                          C
* $$ CLASS=O
// JOB BOESTR04 CATALOG VTAM START OPTION LIST
// EXEC LIBR,PARM='MSHP'
ACCESS SUBLIB=PRD2.CONFIG
CATALOG     ATCSTR04.B        REPLACE=YES
SSCPID=22,                                                           *
HOSTSA=22,                                                           *
SSCPNAME=IPFV2B,                                                     *
HOSTPU=IPFVM22,                                                      *
NETID=DEIBMIPF,                                                      *
MAXSUBA=255,                                                         *
CONFIG=04,                                                           *
IOINT=0,                                                             *
SGALIMIT=1M,                                                         *
BSBUF=(28,,,1),                                                      *
CRPLBUF=(60,,,1),                                                    *
LFBUF=(300,288,,20),             FROM 70 TO 120                      *
LPBUF=(12,,,6),                                                      *
SFBUF=(20,,,20),                                                     *
SPBUF=(210,,,32),                                                    *
VFBUF=204800,                    FROM 102400 TO 122880 (ADD 5P)      *
VPBUF=528384,                    FROM 446464 TO 528384 (ADD 20P)     *
XDBUF=(6,,,1)
/+
/*
/&
* $$ EOJ
```

A.2 VSE Virtual Machine Directory

```
USER V133A80K PASSWORD 0032M 64M G
  ACCOUNT V133A80K V133A80K
  OPTION MAXCON 150
  MACHINE ESA
  IPL CMS
  CONSOLE 0009 3215 T
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  SPECIAL 080 3270
  SPECIAL 081 3270
  SPECIAL 082 3270
  SPECIAL 083 3270
  SPECIAL 084 3270
  SPECIAL 085 3270
  SPECIAL 086 3270
  SPECIAL 087 3270
  SPECIAL 088 3270
```

```
SPECIAL 089 3270
SPECIAL 800 CTCA VTAM
DEDICATE 300 0300
DEDICATE 301 0301
DEDICATE 302 0302
DEDICATE 303 0303
DEDICATE 960 2960
MDISK 191 3380  001  049 DISK01 MW ALL
MDISK 192 3380  001  049 DISK02 MW ALL
LINK MAINT   190 0190 RR
LINK MAINT   19D 019D RR
LINK MAINT   19E 019E RR
MDISK 991 3380 1770  885 DISK01 MW RVSE WVSE MVSE
MDISK 992 3380 1770  885 DISK02 MW RVSE WVSE MVSE
MDISK 993 3380  885  885 DISK01 MW RVSE WVSE MVSE
MDISK 994 3380  885  885 DISK02 MW RVSE WVSE MVSE
MDISK 995 3380   50  835 DISK01 MW RVSE WVSE MVSE
MDISK 996 3380   50  835 DISK02 MW RVSE WVSE MVSE
MDISK 997 3380    0 2655 DISK01 MW RVSE WVSE MVSE
MDISK 998 3380    0 2655 DISK02 MW RVSE WVSE MVSE
```

A.3 Define Programs and Transactions

```
* $$ JOB JNM=DFHPPTEZ,CLASS=A,DISP=D,NTFY=YES
* $$ LST CLASS=Q,DISP=H
// JOB DFHPPTC2 ASSEMBLE
// LIBDEF *,CATALOG=PRD2.CONFIG
// OPTION CATAL,LIST
// EXEC ASSEMBLY
         TITLE 'DFHPPTEZ -- SUPPLIED WITH VSE/ESA'
         PUNCH ' CATALOG DFHPPTEZ.OBJ REP=YES'
         DFHPPT TYPE=INITIAL,SUFFIX=EZ
         SPACE 3
* $$ SLI MEM=CICSPPT.USER,S=MQMUSR1.USER
         SPACE 3
         SPACE 3
         DFHPPT TYPE=FINAL
         END   DFHPPTBA
/*
// EXEC LNKEDT
/*
/&
* $$ EOJ

* $$ JOB JNM=DFHPCTEZ,CLASS=A,DISP=D,NTFY=YES
* $$ LST CLASS=Q,DISP=H
// JOB DFHPCTC2 ASSEMBLE
// LIBDEF *,CATALOG=PRD2.CONFIG
// OPTION CATAL,LIST
// EXEC ASSEMBLY
         TITLE 'DFHPCTEZ -- EZBRIDGE GROUP'
         PUNCH ' CATALOG DFHPCTEZ.OBJ REP=YES'
         DFHPCT TYPE=INITIAL,SUFFIX=EZ
* $$ SLI MEM=CICSPCT.USER,S=MQMUSR1.USER
         SPACE 3
         DFHPCT TYPE=FINAL
         END   DFHPCTBA
/*
```

```
// EXEC LNKEDT
/*
/&
* $$ EOJ

// JOB CSDJOB   JOB TO EXECUTE DFHCSDUP
// DLBL DFHCSD,'CICS.CSD',,VSAM,CAT=VSESPUC
// LIBDEF PHASE,SEARCH=PRD2.CONFIG
*
*
// EXEC DFHCSDUP,SIZE=300K
   MIGRATE TABLE(DFHPCTEZ) TOGROUP(EZPCT)
   MIGRATE TABLE(DFHPPTEZ) TOGROUP(EZPPT)
/*
/&
```

A.4 VSAM Definitions for ezBRIDGE on VSE/ESA

```
* $$ JOB JNM=EZVSAM,DISP=D,CLASS=A
// JOB EZVSAM - DEFINE ALL EZBRIDGE VSAM FILES
// EXEC IDCAMS,SIZE=AUTO
 /*  delete and define the configuration file                    */
     DELETE (MQSERIES.MQFCNFG) CL NOERASE PURGE  -
     CATALOG(UCATEZB)
 DEF CLUSTER(NAME(MQSERIES.MQFCNFG) -
     FILE(MQFCNFG)                          -
     VOL(SYSWK3)                      -
     RECORDS (900 300)                      -
     RECORDSIZE (1024 1024)                 -
     INDEXED                                -
     KEYS(100 0 )                           -
     SHR(2))                                -
     DATA (NAME (MQSERIES.MQFCNFG.DATA) CISZ(4096)) -
     INDEX (NAME (MQSERIES.MQFCNFG.INDEX) CISZ(512)) -
     CATALOG(UCATEZB)
 /*                                                              */
 /*  delete and define the log file                              */
     DELETE (MQSERIES.MQFLOG) CL NOERASE PURGE -
     CATALOG(UCATEZB)
 DEF CLUSTER(NAME(MQSERIES.MQFLOG) -
     FILE(MQFLOG)                           -
     VOL(SYSWK3)                      -
     RECORDS (300 100)                      -
     RECORDSIZE (200 4089)                 -
     INDEXED                                -
     KEYS(52 0)                             -
     SHR(2))                                -
     DATA (NAME (MQSERIES.MQFLOG.DATA) CISZ(4096)) -
     INDEX (NAME (MQSERIES.MQFLOG.INDEX) CISZ(1024)) -
     CATALOG(UCATEZB)
 /*                                                              */
 /*  delete and define the monitor file                          */
     DELETE (MQSERIES.MQFMON) CL NOERASE PURGE -
     CATALOG(UCATEZB)
 DEF CLUSTER(NAME(MQSERIES.MQFMON) -
     FILE(MQFMON)                           -
     VOL(SYSWK3)                      -
     RECORDS (300 100)                      -
```

```
    RECORDSIZE (200 4089)                -
    INDEXED                               -
    KEYS(52 0)                            -
    SHR(2))                               -
    DATA (NAME (MQSERIES.MQFMON.DATA) CISZ(4096)) -
    INDEX (NAME (MQSERIES.MQFMON.INDEX) CISZ(1024)) -
    CATALOG(UCATEZB)
/*                                                              */
/*  delete and define the error file                            */
    DELETE (MQSERIES.MQFERR) CL NOERASE PURGE -
    CATALOG(UCATEZB)
DEF CLUSTER(NAME(MQSERIES.MQFERR) -
    FILE(MQFERR)                          -
    VOL(SYSWK3)                      -
    RECORDS (300 100)                     -
    RECORDSIZE (200 4089)                -
    INDEXED                               -
    KEYS(52 0)                            -
    SHR(2))                               -
    DATA (NAME (MQSERIES.MQFERR.DATA) CISZ(4096)) -
    INDEX (NAME (MQSERIES.MQFERR.INDEX) CISZ(1024)) -
    CATALOG(UCATEZB)
/*                                                              */
/*  delete and define local queue file LQVSE                    */
    DELETE (MQSERIES.LQVSE) CL NOERASE PURGE -
    CATALOG(UCATEZB)
DEF CLUSTER(NAME(MQSERIES.LQVSE) -
    FILE(LQVSE)                           -
    VOL(SYSWK3)                      -
    RECORDS (300 100)                     -
    RECORDSIZE (200 4089)                -
    INDEXED                               -
    KEYS(52 0)                            -
    SHR(2))                               -
    DATA (NAME (MQSERIES.LQVSE.DATA) CISZ(4096)) -
    INDEX (NAME (MQSERIES.LQVSE.INDEX) CISZ(1024)) -
    CATALOG(UCATEZB)
/*                                                              */
/*  delete and define local queue file FROMOS2                  */
    DELETE (MQSERIES.FROMOS2) CL NOERASE PURGE -
    CATALOG(UCATEZB)
DEF CLUSTER(NAME(MQSERIES.FROMOS2) -
    FILE(FROMOSN)                          -
    VOL(SYSWK3)                      -
    RECORDS (300 100)                     -
    RECORDSIZE (200 4089)                -
    INDEXED                               -
    KEYS(52 0)                            -
    SHR(2))                               -
    DATA (NAME (MQSERIES.FROMOS2.DATA) CISZ(4096)) -
    INDEX (NAME (MQSERIES.FROMOS2.INDEX) CISZ(1024)) -
    CATALOG(UCATEZB)
/*                                                              */
/*  delete and define transmission queue file QMOS2             */
    DELETE (MQSERIES.QMOS2) CL NOERASE PURGE -
    CATALOG(UCATEZB)
DEF CLUSTER(NAME(MQSERIES.QMOS2) -
    FILE(XOSN)                            -
    VOL(SYSWK3)                      -
```

```
    RECORDS (300 100)                     -
    RECORDSIZE (200 4089)                -
    INDEXED                               -
    KEYS(52 0)                            -
    SHR(2))                               -
    DATA (NAME (MQSERIES.QMOS2.DATA) CISZ(4096)) -
    INDEX (NAME (MQSERIES.QMOS2.INDEX) CISZ(1024)) -
    CATALOG(UCATEZB)
 /*                                                              */
 /*  delete and define local queue file FROMAIX                  */
    DELETE (MQSERIES.FROMAIX) CL NOERASE PURGE -
    CATALOG(UCATEZB)
 DEF CLUSTER(NAME(MQSERIES.FROMAIX) -
    FILE(FROMAIX)                          -
    VOL(SYSWK3)                       -
    RECORDS (300 100)                     -
    RECORDSIZE (200 4089)                -
    INDEXED                               -
    KEYS(52 0)                            -
    SHR(2))                               -
    DATA (NAME (MQSERIES.FROMAIX.DATA) CISZ(4096)) -
    INDEX (NAME (MQSERIES.FROMAIX.INDEX) CISZ(1024)) -
    CATALOG(UCATEZB)
 /*                                                              */
 /*  delete and define transmission queue file QMAIX             */
    DELETE (MQSERIES.QMAIX) CL NOERASE PURGE -
    CATALOG(UCATEZB)
 DEF CLUSTER(NAME(MQSERIES.QMAIX) -
    FILE(QMAIX)                           -
    VOL(SYSWK3)                       -
    RECORDS (300 100)                     -
    RECORDSIZE (200 4089)                -
    INDEXED                               -
    KEYS(52 0)                            -
    SHR(2))                               -
    DATA (NAME (MQSERIES.QMAIX.DATA) CISZ(4096)) -
    INDEX (NAME (MQSERIES.QMAIX.INDEX) CISZ(1024)) -
    CATALOG(UCATEZB)
 /*                                                              */
/*
/&
* $$ EOJ
```

Appendix B. ezBRIDGE on AIX Sample Definitions

B.1 Sample AIX SNA Services/6000 Profiles

```
*  PROFILE TYPE BEING PRINTED: "CONNECTION"

LU62EZ_CONNECTION:
 type = CONNECTION
 profile_name = LU62EZ
 attachment_profile_name = LU62EZ
 local_lu_profile_name = LU62EZ
 network_name = DEIBMIPF
 remote_lu_name = CICSSA22
 stop_connection_on_inactivity = no
 lu_type = lu6.2
 interface_type = extended
 remote_tpn_list_name = RDEFAULT
 mode_list_name = LU62EZ
 node_verification = no
 inactivity_timeout_value = 0
 notify = no
 parallel_sessions = parallel
 negotiate_session_limits = yes
 security_accepted = none
 conversation_security_access_list_name = CONVDEFAULT


*  PROFILE TYPE BEING PRINTED: "LOCAL LU"

LU62EZ_LOCALLU:
 type = LOCALLU
 profile_name = LU62EZ
 local_lu_name = QMAIX
 network_name = DEIBMIPF
 lu_type = lu6.2
 independent_lu = yes
 tpn_list_name = VTPN
 local_lu_address = 1
 sscp_id =
 number_of_rows = 24
 number_of_columns = 80
```

```
*  PROFILE TYPE BEING PRINTED: "ATTACHMENT"

LU62EZ_ATTACHMENT:
 type = ATTACHMENT
 profile_name = LU62EZ
 control_point_profile_name = LU62EZ
 logical_link_profile_name = LU62EZ
 physical_link_profile_name = LU62EZ
 logical_link_type = token_ring
 restart_on_deactivation = yes
 stop_attachment_on_inactivity = no
 station_type = secondary
 physical_link_type = token_ring
 remote_secondary_station_address = 1
 smart_modem_command_sequence =
 length_of_command_sequence = 0
 call_type = call
 x25_level = 1984
 listen_name = IBMQLLC
 autolisten = yes
 timeout_value = 5
 remote_link_name_ethernet =
 remote_link_name_token_ring =
 remote_link_address = 400020201003
 selection_sequence =
 length_of_selection_sequence = 0
 network_type = switched
 access_routing = link_address
 remote_sap_address = 04
 remote_sap_address_range_lower = 04
 remote_sap_address_range_upper = EC
 virtual_circuit_type = permanent
 remote_station_X.25_address =
 optional_X.25_facilities = no
 logical_channel_number_of_PVC = 1
 reverse_charging = no
 rpoa = no
 default_packet_size = no
 default_window_size = no
 default_throughput_class = no
 closed_user_group = no
 closed_user_group_outgoing = no
 network_user_id = no
 network_user_id_name =
 data_network_identification_code =
 packet_size_for_received_data = 128
 packet_size_for_transmit_data = 128
 window_size_for_received_data = 2
 window_size_for_transmit_data = 2
 throughput_class_for_received_data = 9600
 throughput_class_for_transmit_data = 9600
 index_to_selected_closed_user_group = 0
 lu_address_registration = no
 lu_address_registration_name = LDEFAULT
```

```
*  PROFILE TYPE BEING PRINTED: "SNA"

sna_SNA:
 type = SNA
 profile_name = sna
 total_active_open_connections = 200
 total_sessions = 200
 total_conversations = 200
 server_synonym_name = sna
 nmvt_action_when_no_nmvt_process = reject
 restart_action = once
 stdin = /dev/null
 stdout = /dev/console
 stderr = /dev/console
 sna_error_log = no



*  PROFILE TYPE BEING PRINTED: "REMOTE TPN"

MQO1_REMOTETPN:
 type = REMOTETPN
 profile_name = MQO1
 tpn_name = MQO1
 tpn_name_hex = D4D8FOF1
 pip_data = no
 conversation_type = mapped
 recovery_level = no_reconnect
 sync_level = none
 tpn_name_in_hex = no



*  PROFILE TYPE BEING PRINTED: "REMOTE TPN LIST"

RDEFAULT_REMOTETPNLIST:
 type = REMOTETPNLIST
 Listname = RDEFAULT
 list_members = RDEFAULT,MQO1


*  PROFILE TYPE BEING PRINTED: "TPN"

VTPN_TPN:
 type = TPN
 profile_name = VTPN
 tpn_name =
 tpn_name_hex =
 conversation_type = mapped
 pip_data = no
 sync_level = either
 recovery_level = no_reconnect
 full_path_to_tpn_executable = /usr/lpp/sna
 multiple_instances = no
 user_id = 0
 server_synonym_name =
 restart_action = once
 communication_type = signals
 stdin = /dev/null
```

```
 stdout = /dev/console
 stderr = /dev/console
 subfields = 0
 communication_ipc_queue_key = 0
 tpn_name_in_hex = no
 security_required = none
 resource_security_access_list_name = RSRCDEFAULT



*  PROFILE TYPE BEING PRINTED: "TPN LIST"

VTPN_TPNLIST:
 type = TPNLIST
 Listname = VTPN
 list_members = VTPN




*  PROFILE TYPE BEING PRINTED: "CONTROL POINT"

LU62EZ_CONTROLPOINT:
 type = CONTROLPOINT
 profile_name = LU62EZ
 xid_node_id = 071E0009
 network_name = DEIBMIPF
 cp_name = IPFCPX11



*  PROFILE TYPE BEING PRINTED: "MODE"

LU62PS_MODE:
 type = MODE
 profile_name = LU62PS
 mode_name = LU62PS
 maximum_number_of_sessions = 8
 minimum_contention_winners = 4
 minimum_contention_losers = 4
 auto_activations_limit = 0
 receive_pacing = 3
 send_pacing = 3
 maximum_ru_size = 2816
 recovery_level = no_reconnect



*  PROFILE TYPE BEING PRINTED: "MODE LIST"

LU62EZ_MODELIST:
 type = MODELIST
 Listname = LU62EZ
 list_members = LU62PS
```

```
*  PROFILE TYPE BEING PRINTED: "TOKEN RING LOGICAL"

LU62EZ_TOKENRINGLOGICAL:
 type = TOKENRINGLOGICAL
 profile_name = LU62EZ
 retry_limit = 20
 transmit_window_count = 10
 dynamic_window_increment = 1
 retransmit_count = 8
 receive_window_count = 127
 ring_access_priority = 0
 inactivity_timeout = 48
 drop_link_on_inactivity = yes
 response_timeout = 2
 acknowledgement_timeout = 1
 force_disconnect_timeout = 120
 link_trace = no
 trace_entry_size = short
 logical_link_type = token_ring
 maximum_i_field = system_defined
 maximum_i_field_size = 30729
 physical_link_type = token_ring



*  PROFILE TYPE BEING PRINTED: "TOKEN RING PHYSICAL"

LU62EZ_TOKENRINGPHYSICAL:
 type = TOKENRINGPHYSICAL
 profile_name = LU62EZ
 device_name = tok0
 local_link_name =
 local_sap_address = 04
 physical_link_type = token_ring
 maximum_number_of_logical_links = 32



*  PROFILE TYPE BEING PRINTED: "LU REGISTRATION"

LDEFAULT_LUREGISTRATION:
 type = LUREGISTRATION
 profile_name = LDEFAULT
 local_lu_addresses = 64



*  PROFILE TYPE BEING PRINTED: "CONV LIST"

CONVDEFAULT_CONVLIST:
 type = CONVLIST
 Listname = CONVDEFAULT
 list_members =
```

```
*  PROFILE TYPE BEING PRINTED: ”RSRC LIST”

RSRCDEFAULT_RSRCLIST:
 type = RSRCLIST
 Listname = RSRCDEFAULT
 list_members =


*  PROFILE TYPE BEING PRINTED: ”CPI-C SIDE INFORMATION”

WDEFAULT_CPICSIDE:
 type = CPICSIDE
 profile_name = WDEFAULT
 partner_lu_name = CDEFAULT
 remote_TP_name =
 rtpn_name_hex =
 service_TP = no
 mode_name =
```

Appendix C. ezBRIDGE Communication Definitions Summary

The tables provided in this appendix summarize the ezBRIDGE resource definitions we made for all three ezBRIDGE platforms used in our MQI environment, ezBRIDGE on VSE/ESA, ezBRIDGE on AIX and ezBRIDGE on OS/2. For cross-checking these resources refer to Figure 4 on page 17 and Figure 81 on page 102.

The queues we used for local Installation Verification Tests are not included.

C.1 Queue Manager QMVSE on VSE/ESA

Resource type	Resource name	Description
queue	QMAIX	local transmission queue
	TOAIX	remote queue
	FROMAIX	local queue
	QMOS2S	local transmission queue
	TOOS2	remote queue
	FROMOS2	local queue
	TOOS22	remote queue
channel	CVSEAIX	sender channel
	CAIXVSE	receiver channel
	CVSEOS2	sender channel
	COS2VSE	receiver channel

Table 3. ezBRIDGE on VSE/ESA Customization Summary

C.2 Queue Manager QMAIX on AIX

Resource type	Resource name	Description
queue	QMVSE	local transmission queue
	TOVSE	remote queue
	FROMVSE	local queue
channel	CAIXVSE	sender channel
	CVSEAIX	receiver channel

Table 4. ezBRIDGE on AIX Customization Summary

C.3 Queue Manager QMOS2S on Workstation1 (OS2S0)

Resource type	Resource name	Description
queue	QMVSE	local transmission queue
	TOVSE	remote queue
	FROMVSE	local queue
	QMOS22	local transmission queue
	TOOS22	remote queue
	FROMOS22	local queue
channel	COS2VSE	sender channel
	CVSEOS2	receiver channel
	COS2SOS22	sender channel
	COS22OS2S	receiver channel

Table 5. Customization Summary for Workstation 1 (OS2S)

C.4 Queue Manager QMOS22 on Workstation 2 (OS22)

Resource type	Resource name	Description
queue	QMOS2S	local transmission queue
	TOOS2S	remote queue
	FROMOS2S	local queue
	TOVSE	remote queue
	FROMVSE	local queue
channel	COS22OS2S	sender channel
	COS2SOS22	receiver channel

Table 6. Customization Summary for Workstation 2 (OS22)

C.5 Message Channel and MCA Summary

The table below summarizes the definitions of the message channel and their corresponding transaction programs or names.

Queue Manager	Sender Channel	Sender MCA	Receiver Channel	Receiver MCA
QMVSE	CVSEOS2	MQPSEND	COS2VSE	MQ01
	CVSEAIX	MQPSEND	CAIXVSE	MQ01
QMOS2S	COS2VSE	TPSV (Note)	CVSEOS2	TPVS (Note)
	COS2SOS22	TPS2 (Note)	COS22OS2S	TP2S (Note)
QMOS22	COS22OS2S	TP2S (Note)	COS2SOS22	TPS2 (Note)
QMAIX	CAIXVSE	MQ01	CVSEAIX	VTPN
Note: Name of the corresponding CM/2 definition to identify the MCA transaction program.				

Table 7. Message Channel and MCA Summary

List of Abbreviations

APA	all points addressable
APPC	Advanced Program to Program Communications
ASCII	American National Standard for Information Interchange
BMS	Basic Mapping Support
CM/2	Communication Manager/2
DLC	Data Link Control
DPL	Distributed Program Link
DTP	Distributed Transaction Processing
EBCDIC	Extended Binary-Coded Decimal Interchange Code
FS	Function Shipping
GCS	Group Control System
GUI	Graphical User Interface
IBM	International Business Machines Corporation
ISC	Inter System Communication
ITSO	International Technical Support Organization
IVT	Installation Verification Test
LAPS	LAN Adapter and Protocol Support
LPP	Licensed Program Product
LU	Logical Unit
MAC	Media Access Control
MCA	Message Channel Agent
MCP	Message Channel Protocol
MQI	Message Queue Interface
MQM	Message Queue Manager
NetBIOS	Network Basic Input Output System
NFS	Network File System
PTF	Program Temporary Fix
PD	Program Definition
PROFS*	Professional Office System
RTPN	Remote Transaction Program Name
RD	Region Definition
RPC	Remote Procedure Call
SFS	Structured File Server
SMIT	System Management Interface Tool
SSCP	System Services Control Point
TCP/IP	Transmission Control Protocol/Internet Protocol
TD	Transaction Definition
TPN	Transaction Program Name
TR	Transaction Routing
VSAM	Virtual Storage Access Method
XCA	External Communication Adapter

Index

Numerics

A

C

D

E

G

I

L

M

O

Q

R

S

T

V

X

ITSO Technical Bulletin Evaluation RED000

CICS/VSE Client/Server Solutions
Implementing the Message Queue Interface

Publication No. GG24-4263-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

ITSO Technical Bulletin Evaluation **RED000**
GG24-4263-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM International Technical Support Organization
Department 3222, Building 71032-02
POSTFACH 1380
71032 BOEBLINGEN
GERMANY

Fold and Tape **Please do not staple** Fold and Tape

GG24-4263-00

Cut or Fold
Along Line

Printed in U.S.A.